50 Interview Question on Web Back-End Development

1. What is your favorite programming language and why?

Answer: I love JavaScript. JavaScript is simple to comprehend and pick up. Both users and developers will find the structure to be straightforward. Additionally, it is very doable to implement, saving web developers a tonne of money when creating dynamic content. JavaScript is a "interpreted" language, it cuts down on the time needed for compilation in other programming languages like Java. Another client-side script is JavaScript, which accelerates programme execution by eliminating the wait time for server connections.

No matter where JavaScript is hosted, it is always run in a client environment to reduce bandwidth usage and speed up execution.

That's why I like JavaScript and I work with it regular.

2. What is Node.js? Where can you use it?

Answer: Node.js is an open-source, cross-platform JavaScript runtime environment and library to run web applications outside the client's browser. It is used to create server-side web applications.

Node.js is perfect for data-intensive applications as it uses an asynchronous, event-driven model. You can use I/O intensive web applications like video streaming sites. You can also use it for developing: Real-time web applications, Network applications, General-purpose applications, and Distributed systems.

3. Why use Node.js?

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

- It is generally fast
- It rarely blocks
- It offers a unified programming language and data type
- Everything is asynchronous
- It yields great concurrency

4. How does Node.js work?

Answer: A web server using Node.js typically has a workflow that is quite similar to the diagram illustrated below. Let's explore this flow of operations in detail.

- Clients send requests to the webserver to interact with the web application. Requests can be non-blocking or blocking:
- Querying for data
- Deleting data
- Updating the data
- Node.js retrieves the incoming requests and adds those to the Event Queue
- The requests are then passed one-by-one through the Event Loop. It checks if the requests are simple enough not to require any external resources
- The Event Loop processes simple requests (non-blocking operations), such as I/O Polling, and returns the responses to the corresponding clients

A single thread from the Thread Pool is assigned to a single complex request. This thread is responsible for completing a particular blocking request by accessing external resources, such as computation, database, file system, etc.

Once the task is carried out completely, the response is sent to the Event Loop that sends that response back to the client.

5. Why is Node.js Single-threaded?

Answer: 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 instead of the typical thread-based implementation.

6. If Node.js is single-threaded, then how does it handle concurrency?

Answer: The Multi-Threaded Request/Response Stateless Model is not followed by the Node JS Platform, and it adheres to the Single-Threaded Event Loop Model. The Node JS Processing paradigm is heavily influenced by the JavaScript Event-based model and the JavaScript callback system. As a result, Node.js can easily manage more concurrent client requests. The event loop is the processing model's beating heart in Node.js.

7. Explain callback in Node.js.

Answer: A callback function is called after a given task. It 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 are the advantages of using promises instead of callbacks?

Answer:

- The control flow of asynchronous logic is more specified and structured.
- The coupling is low.
- We've built-in error handling.
- Improved readability.

9. How would you define the term I/O?

Answer:

- The term I/O is used to describe any program, operation, or device that transfers data to or from a medium and to or from another medium
- Every transfer is an output from one medium and an input into another. The medium can be a physical device, network, or files within a system

10. How is Node.js most frequently used?

Answer: Node.js is widely used in the following applications:

5. Real-time chats
6. Internet of Things
7. Complex SPAs (Single-Page Applications)
8. Real-time collaboration tools
9. Streaming applications
10. Microservices architecture

11. Explain the difference between frontend and backend development?

Answer:

Front-end	Back-end
Frontend refers to the client-side of an application	Backend refers to the server-side of an application
It is the part of a web application that users can see and interact with	It constitutes everything that happens behind the scenes
It typically includes everything that attributes to the visual aspects of a web application	It generally includes a web server that communicates with a database to serve requests
HTML, CSS, JavaScript, AngularJS, and ReactJS are some of the essentials of frontend development	Java, PHP, Python, and Node.js are some of the backend development technologies

12. What is NPM?

Answer: NPM stands for Node Package Manager, responsible for managing all the packages and modules for Node.js.

Node Package Manager provides two main functionalities:

- Provides online repositories for node.js packages/modules, which are searchable on search.nodejs.org
- Provides command-line utility to install Node.js packages and also manages Node.js versions and dependencies

13. What are the modules in Node.js?

Answer: Modules are like JavaScript libraries that can be used in a Node.js application to include a set of functions. To include a module in a Node.js application, use the require() function with the parentheses containing the module's name.

Node.js has many modules to provide the basic functionality needed for a web application. Some of them include:

Core Modules	Description
HTTP	Includes classes, methods, and events to create a Node.js HTTP server
util	Includes utility functions useful for developers
fs	Includes events, classes, and methods to deal with file I/O operations
url	Includes methods for URL parsing
query string	Includes methods to work with query string
stream	Includes methods to handle streaming data
zlib	Includes methods to compress or decompress files

14. What is the purpose of the module .Exports?

Answer: In Node.js, a module encapsulates all related codes into a single unit of code that can be parsed by moving all relevant functions into a single file. You may export a module with the module and export the function, which lets it be imported into another file with a needed keyword.

15. Why is Node.js preferred over other backend technologies like Java and PHP?

Answer: Some of the reasons why Node.js is preferred include:

- Node.js is very fast
- Node Package Manager has over 50,000 bundles available at the developer’s disposal
- Perfect for data-intensive, real-time web applications, as Node.js never waits for an API to return data
- Better synchronization of code between server and client due to same code base
- Easy for web developers to start using Node.js in their projects as it is a JavaScript library

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

Answer:

Angular	Node.js
It is a frontend development framework	It is a server-side environment
It is written in TypeScript	It is written in C, C++ languages
Used for building single-page, client-side web applications	Used for building fast and scalable server-side networking applications
Splits a web application into MVC components	Generates database queries

17. Which database is more popularly used with Node.js?

Answer: MongoDB is the most common database used with Node.js. It is a NoSQL, cross-platform, document-oriented database that provides high performance, high availability, and easy scalability.

18. What are some of the most commonly used libraries in Node.js?

Answer: There are two commonly used libraries in Node.js:

- ExpressJS - Express is a flexible Node.js web application framework that provides a wide set of features to develop web and mobile applications.
- Mongoose - Mongoose is also a Node.js web application framework that makes it easy to connect an application to a database.

19. What are the pros and cons of Node.js?

Answer:

Node.js Pros	Node.js Cons
Fast processing and an event-based model	Not suitable for heavy computational tasks

Uses JavaScript, which is well-known amongst developers	Using callback is complex since you end up with several nested callbacks
Node Package Manager has over 50,000 packages that provide the functionality to an application	Dealing with relational databases is not a good option for Node.js
Best suited for streaming huge amounts of data and I/O intensive operations	Since Node.js is single-threaded, CPU intensive tasks are not its strong suit

20. What is the command used to import external libraries?

Answer: The “require” command is used for importing external libraries. For example - “var http=require (“HTTP”).” This will load the HTTP library and the single exported object through the HTTP variable.

Now that we have covered some of the important beginner-level Node.js interview questions let us look at some of the intermediate-level Node.js interview questions.

21. What is an Event Loop in Node.js?

Answer: Event loops handle asynchronous callbacks in Node.js. It is the foundation of the non-blocking input/output in Node.js, making it one of the most important environmental features.

22. What are the two types of API functions in Node.js?

Answer: The two types of API functions in Node.js are:

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

23. What is the package.json file?

Answer: The package.json file is the heart of a Node.js system. This file holds the metadata for a particular project. The package.json file is found in the root directory of any Node application or module

This is what a package.json file looks like immediately after creating a Node.js project using the command: npm init

You can edit the parameters when you create a Node.js project.

24. How would you use a URL module in Node.js?

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

25. What is the Express.js package?

Answer: Express is a flexible Node.js web application framework that provides a wide set of features to develop both web and mobile applications.

26. How do you create a simple Express.js application?

Answer:

- The request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on
- The response object represents the HTTP response that an Express app sends when it receives an HTTP request

27. What are streams in Node.js?

Answer: Streams are objects that enable you to read data or write data continuously.

There are four types of streams:

Readable – Used for reading operations

Writable − Used for write operations

Duplex − Can be used for both reading and write operations

Transform − A type of duplex stream where the output is computed based on input

28. How do you install, update, and delete a dependency?

Answer:

29. How do you create a simple server in Node.js that returns Hello World?

Answer:

- Import the HTTP module
- Use createServer function with a callback function using request and response as parameters.
- Type “hello world."
- Set the server to listen to port 8080 and assign an IP address

30. Explain asynchronous and non-blocking APIs in Node.js.

Answer:

- All Node.js library APIs are asynchronous, which means they are also non-blocking
- A Node.js-based server never waits for an API to return data. Instead, it moves to the next API after calling it, and a notification mechanism from a Node.js event responds to the server for the previous API call

31. What is a callback function in Node.js?

Answer: A callback is a function called after a given task. This prevents any blocking and enables other code to run in the meantime.

In the last section, we will now cover some of the advanced-level Node.js interview questions.

32. What is callback hell?

Answer:

- Callback hell, also known as the pyramid of doom, is the result of intensively nested, unreadable, and unmanageable callbacks, which in turn makes the code harder to read and debug
- improper implementation of the asynchronous logic causes callback hell

33. For Node.js, why does Google use the V8 engine?

Answer: The V8 engine, developed by Google, is open-source and written in C++. Google Chrome makes use of this engine. V8, unlike the other engines, is also utilized for the popular Node.js runtime. V8 was initially intended to improve the speed of JavaScript execution within web browsers. Instead of employing an interpreter, V8 converts JavaScript code into more efficient machine code to increase performance. It turns JavaScript code into machine code during execution by utilizing a JIT (Just-In-Time) compiler, as do many current JavaScript engines such as SpiderMonkey or Rhino (Mozilla).

34. Explain the concept of middleware in Node.js.

Answer: Middleware is a function that receives the request and response objects. Most tasks that the middleware functions perform are:

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

35. What are the different types of HTTP requests?

Answer: HTTP defines a set of request methods used to perform desired actions. The request methods include:

GET: Used to retrieve the data

POST: Generally used to make a change in state or reactions on the server

HEAD: Similar to the GET method, but asks for the response without the response body

DELETE: Used to delete the predetermined resource

36. How would you connect a MongoDB database to Node.js?

Answer: To create a database in MongoDB:

- Start by creating a MongoClient object
- Specify a connection URL with the correct IP address and the name of the database you want to create

---

37. What is Express.js?

Answer: Express.js, or simply Express, is a free, open-source, lightweight, and fast backend web application framework for Node.js. It is released as open-source software under the MIT License.

It is designed for building single-page, multi-page, and hybrid web applications and APIs. It is called the de facto standard server framework for Node.js. It was founded and developed by TJ Holowaychuk in 2010 and written in JavaScript.

38. What are some distinctive features of Express?

Answer: As Express is a lightweight, minimal and flexible Node.js web application framework, it provides a robust set of features for web and mobile applications. Following is the list of some distinctive features of this framework:

- js can be used to design single-page, multi-page, and hybrid web applications and APIs.
- It allows to set up middleware to respond to HTTP/RESTful Requests.
- It defines a routing table to perform different HTTP operations (method and URL).

- It allows to dynamically rendering HTML Pages based on passing arguments to templates.
- It provides high performance because of its ultra-fast I/O. It prepares a thin layer; therefore, the performance is adequate.
- Its MVC-like structure makes it organize the web application into MVC architecture.
- It provides good database support. It supports RDBMS as well as NoSQL databases.
- It is asynchronous and single-threaded.
- Its robust API makes routing easy.

39. Is Express.js front-end or backend framework?

Answer: Express.js or Express is a JavaScript backend framework. It is mainly designed to develop complete web applications (single-page, multi-page, and hybrid web applications) and APIs. Express is the backend component of the MEAN stack where M stands for MongoDB, which handles database; E stands for Express, which handles backend; A stands for AngularJS, which is for the front-end, and N stands for Node.

40. Why do we use Express.js?

Answer: Express.js is an automatically prebuilt Node.js framework that facilitates us to create server-side web applications faster and smarter. The main reason for choosing Express is its simplicity, minimalism, flexibility, and scalability characteristics.

41. What is the difference between Express.js and Node.js?

Answer: Node.js is an open-source, cross-platform run-time environment used for executing JavaScript code outside of a browser. Node.js is not a framework or a programming language; it is a platform that acts as a web server. Many big companies such as Paypal, Uber, Netflix, Wallmart, etc., are using this. On the other hand, Express is a small framework based on the functionality of Node.js.

Some key differences between Express.js and Node.js:

Feature	Express.js	Node.js

Definition	Express.js is a lightweight and fast backend web application framework for Node.js.	Node.js is an open-source and cross-platform that is used to execute JavaScript code outside of a browser.
Usage	Express.js is used to develop complete web applications such as single-page, multi-page, and hybrid web applications and APIs. It uses approaches and principles of Node.js.	Node.js is used to build server-side, input-output, event-driven apps.
Features	Express has more features than Node.js.	Node.js has fewer features as compared to Express.js.
Building Block	Express.js is built on Node.js.	Node.js is built on Google's V8 engine.
Written in	Express.js is written in JavaScript only.	Node.js is written in C, C++, and JavaScript language.
Framework/Platform	Express.js is a framework of Node.js based on its functionalities.	Node.js is a run-time platform or environment designed for server-side execution of JavaScript.
Controllers	Express.js is assigned with controllers.	Node.js is assigned with controllers.
Routing	Routing is provided in Express.js.	Routing is not provided in Node.js.
Middleware	Express.js uses middleware to arrange the functions systematically on the server-side.	Node.js doesn't use any such provision of middleware.

Coding	Express is easy to code and requires less coding time.	Node.js requires more coding time as compare to Express.js.

42. How does an Express code look like?

Answer: The express.js program is saved with ".js" extension.

See the example:

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

var app = express();

app.get('/', function (req, res) {

res.send('Welcome to JavaTpoint!');

});

var server = app.listen(8000, function () {

var host = server.address().address;

var port = server.address().port;

console.log('Example app listening at http://%s:%s', host, port);

});
```

When you run the Node.js command prompt, the app will listen at the specified server address and give the following output.

Output:

```
Welcome to JavaTpoint!
```

43. Write a code to get post a query in Express.js.

Answer:

```
var bodyParser = require('body-parser')

app.use( bodyParser.json() );       // to support JSON-encoded
```

```
app.use(bodyParser.urlencoded({     // to support URL-encoded
 extended: true
}));
```

44. Which are the arguments available to an Express JS route handler function?

Answer: Following are the arguments that are available to an Express.js route handler-function:

- **Req:** the request object
- **Res:** the response object
- **Next (optional):** It is a function employed to pass management to one of the above route handlers.

45. What is the difference between Express and Django?

Answer: Django is a standalone and lightweight web server for testing and development. On the other hand, Express.js is a Node.js framework that can set the middleware to reply to any HTTP request.

Following is a list of some key differences between Express.js and Django:

Aspects	Express.js	Django
Architecture	Express follows the MVC architecture.	Django supports the MTV (Model Template View) design. It uses managing data for interacting and validating.
Framework	Express is a free, open-source, lightweight, and fast backend web application framework for Node.js to build single-page, multi-page, and hybrid web applications and APIs.	This is a Python-based framework used to develop computer apps in a specified time frame.

Efficiency	It is best for developing web applications rapidly on Node.js.	It is more efficient and delivers at a fast speed so, it is cost-effective.
Programming language	The Express framework is programmed in Node.js.	Django is programmed in Python programming language.
Complexity	Express.js is less complex than Django.	Django is more complex than Express.js
Scalability	It provides better scalability.	It is less scalable.
Flexibility	Express is a flexible, minimal API-developing Node.js tool.	It provides limited flexibility.
Full-stack development	It provides a full-stack development that reduces the cost as you don't need to hire several developers to administer a web application's backend and frontend.	It does not deliver full-stack development.
Companies using this technology	Companies such as PayPal, IBM, Fox Sports, etc., are using this technology.	Companies such as Instagram, Mozilla, Bitbucket, etc., are using this technology.

46. How can you enable debugging in Express.js app?

Answer: There are different ways to enable debugging in Express.js app in different Operating Systems

Use the following command on Linux:

```
DEBUG=express:*

node app.js
```

Use the following command on Windows:

```
set DEBUG=express:*
```

```
node app.js
```

47. How can you allow CORS in Express.js?

Answer: We can allow CORS in Express.js, by adding the following code in server.js:

```
app.all('*', function(req, res, next) {

res.set('Access-Control-Allow-Origin', '*');

res.set('Access-Control-Allow-Methods', 'GET, POST, DELETE, PUT');

res.set('Access-Control-Allow-Headers', 'X-Requested-With, Content-Type');

if ('OPTIONS' == req.method) return res.send(200);

next();

});
```

48. How can you deal with error handling in Express.js? Explain with an example.

Answer: Error handling is much easier in the Express versions over Express 4.0. Use the following steps to do the error handling:

Create an Express.js application. There is no built-in middleware like error handler in express 4.0, so you have to either install a middleware or create a custom one.

Create a Middleware:

Create a middleware as following:

```
// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page
```

```
res.status(err.status || 500);
res.render('error');
});
```

Install Error Handler Middleware:

Install the errorhandler as following:

```
npm install errorhandler --save
```

Create a variable:

```
var errorhandler = require('errorhandler')
```

Use the middleware as following:

```
if (process.env.NODE_ENV === 'development') {
 // only use in development
  app.use(errorhandler({log: errorNotification}))
}
function errorNotification(err, str, req) {
 var title = 'Error in ' + req.method + ' ' + req.url
 notifier.notify({
   title: title,
   message: str
 })
}
```

49. What is Middleware in Express.js? What are the different types of Middleware?

Answer: Middleware is a function invoked by the Express routing layer before the final request handler.

Middleware functions are used to perform the following tasks:

- It is used to execute any code.
- It is also used to make changes to the request and the response objects.
- It is responsible for ending the request-response cycle.
- It can call the next middleware function in the stack.

Type of Middleware

Following are the main types of Middleware:

- Application-level Middleware
- Router-level Middleware
- Error-handling Middleware
- Built-in Middleware
- Third-party Middleware

Application-level middleware: The application-level middleware method is used to bind to the app object using app.use() method. It applies on all routes.

```
//This middleware will execute for each route.

app.use(function (req, res, next) {

console.log('Current Time:', Date.now())

next()

})
```

Router-level Middleware: The router-level Middleware is used to bind to a specific instance of express.Router().Built-in Middleware: The built-in Middleware was introduced with version 4.x. It ends the dependency on Connect.

There are the following built-in middleware functions in Express.js:

- **static:** It is used to serve static assets such as HTML files, images, etc.
- **json:** It is used to parse the incoming requests with JSON payloads. It is available with Express 4.16.0+
- **urlencoded:** It is used to parse the incoming requests with URL-encoded payloads. It is available with Express 4.16.0+

Third-party Middleware: There are many third-party middleware available such as:

- Body-parser

- Cookie-parser
- Mongoose
- Sequelize
- Cors
- Express-validator

To handle HTTP POST requests in Express.js version 4 and above, we have to install a middleware module called body-parser. Body-parser extracts the entire body portion of an incoming request stream and exposes it on req.body, The Middleware was a part of Express.js earlier, but now you have to install it separately. You can install it by using the following command:

```
npm install MODULE_NAME
```

You can load it by using requires and used later:

See the Example:

```
var bodyParser = require('body-parser');

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }))
```

See the Example:

```
var middlewareArray = [middleware1, middleware2]

app.get('/home', middlewareArray, function (req, res, next) {

  //Code snippets

})
```

50. What do you understand by NoSQL databases? Is MongoDB a NoSQL database? explain.

Answer: At the present time, the internet is loaded with big data, big users, big complexity etc. and also becoming more complex day by day. NoSQL is answer of all these problems, It is not a traditional database management system, not even a relational database management system (RDBMS). NoSQL stands for "Not Only SQL". NoSQL is a type of database that can handle and sort all type of unstructured, messy and complicated data. It is just a new way to think about the database.

Yes. MongoDB is a NoSQL database.

51. Which are the different languages supported by MongoDB?

Answer: MonggoDB provides official driver support for C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go and Erlang.

You can use MongoDB with any of the above languages. There are some other community supported drivers too but the above mentioned ones are officially provided by MongoDB.

52. What are the different types of NoSQL databases? Give some example.

Answer: NoSQL database can be classified as 4 basic types:

11. Key value store NoSQL database
12. Document store NoSQL database
13. Column store NoSQL database
14. Graph base NoSQL databse

There are many NoSQL databases. MongoDB, Cassandra, CouchBD, Hypertable, Redis, Riak, Neo4j, HBASE, Couchbase, MemcacheDB, Voldemort, RevenDB etc. are the examples of NoSQL databases.

53. Is MongoDB better than other SQL databases? If yes then how?

Answer: MongoDB is better than other SQL databases because it allows a highly flexible and scalable document structure.

For example:

- One data document in MongoDB can have five columns and the other one in the same collection can have ten columns.
- MongoDB database are faster than SQL databases due to efficient indexing and storage techniques.

54. What type of DBMS is MongoDB?

Answer: MongoDB is a document oriented DBMS

55. Why MongoDB is known as best NoSQL database?

Answer: MongoDb is the best NoSQL database because, it is:

- ✓ Document Oriented
- ✓ Rich Query language
- ✓ High Performance
- ✓ Highly Available
- ✓ Easily Scalable

56. Does MongoDB support primary-key, foreign-key relationship?

Answer: No. By Default, MongoDB doesn't support primary key-foreign key relationship.

57. Can you achieve primary key - foreign key relationships in MongoDB?

Answer: We can achieve primary key-foreign key relationship by embedding one document inside another. For example: An address document can be embedded inside customer document.

58. Does MongoDB need a lot of RAM?

Answer: No. There is no need a lot of RAM to run MongoDB. It can be run even on a small amount of RAM because it dynamically allocates and de-allocates RAM according to the requirement of the processes.

59. Explain the structure of ObjectID in MongoDB.

Answer: ObjectID is a 12-byte BSON type. These are:

- ✓ 4 bytes value representing seconds
- ✓ 3 byte machine identifier
- ✓ 2 byte process id
- ✓ 3 byte counter

60. Is it true that MongoDB uses BSON to represent document structure?

Answer: Yes.

61. What are Indexes in MongoDB?

Answer: In MondoDB, Indexes are used to execute query efficiently. Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to

select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

62. By default, which index is created by MongoDB for every collection?

Answer: By default, the_id collection is created for every collection by MongoDB.

63. What is a Namespace in MongoDB?

Answer: Namespace is a concatenation of the database name and the collection name. Collection, in which MongoDB stores BSON objects.

64. Can journaling features be used to perform safe hot backups?

Answer: Yes.

65. Why does Profiler use in MongoDB?

Answer: MongoDB uses a database profiler to perform characteristics of each operation against the database. You can use a profiler to find queries and write operations

66. If you remove an object attribute, is it deleted from the database?

Answer: Yes, it be. Remove the attribute and then re-save() the object.

66. In which language MongoDB is written?

Answer: MongoDB is written and implemented in C++.

67. Does MongoDB need a lot space of Random Access Memory (RAM)?

Answer: No. MongoDB can be run on small free space of RAM.

68. What language you can use with MongoDB?

Answer: MongoDB client drivers supports all the popular programming languages so there is no issue of language, you can use any language that you want.

69. Does MongoDB database have tables for storing records?

Answer: No. Instead of tables, MongoDB uses "Collections" to store data.

70. Do the MongoDB databases have schema?

Answer: Yes. MongoDB databases have dynamic schema. There is no need to define the structure to create collections.

71. What is the method to configure the cache size in MongoDB?

Answer: MongoDB's cache is not configurable. Actually MongoDb uses all the free spaces on the system automatically by way of memory mapped files.

72. How to do Transaction/locking in MongoDB?

Answer: MongoDB doesn't use traditional locking or complex transaction with Rollback. MongoDB is designed to be light weighted, fast and predictable to its performance. It keeps transaction support simple to enhance performance.

73. Why 32 bit version of MongoDB are not preferred ?

Answer: Because MongoDB uses memory mapped files so when you run a 32-bit build of MongoDB, the total storage size of server is 2 GB. But when you run a 64-bit build of MongoDB, this provides virtually unlimited storage size. So 64-bit is preferred over 32-bit.

74. Is it possible to remove old files in the moveChunk directory?

Answer: Yes, These files can be deleted once the operations are done because these files are made as backups during normal shard balancing operation. This is a manual cleanup process and necessary to free up space.

75. What will have to do if a shard is down or slow and you do a query?

Answer: If a shard is down and you even do query then your query will be returned with an error unless you set a partial query option. But if a shard is slow them Mongos will wait for them till response.

76. Explain the covered query in MongoDB.

Answer: A query is called covered query if satisfies the following two conditions:

- The fields used in the query are part of an index used in the query.
- The fields returned in the results are in the same index.

77. What is the importance of covered query?

Answer: Covered query makes the execution of the query faster because indexes are stored in RAM or sequentially located on disk. It makes the execution of the query faster.

Covered query makes the fields are covered in the index itself, MongoDB can match the query condition as well as return the result fields using the same index without looking inside the documents.

78. What is sharding in MongoDB?

Answer: In MongoDB, Sharding is a procedure of storing data records across multiple machines. It is a MongoDB approach to meet the demands of data growth. It creates horizontal partition of data in a database or search engine. Each partition is referred as shard or database shard.

79. What is replica set in MongoDB?

Answer: A replica can be specified as a group of mongo instances that host the same data set. In a replica set, one node is primary, and another is secondary. All data is replicated from primary to secondary nodes.

80. What is primary and secondary replica set in MongoDB?

Answer: In MongoDB, primary nodes are the node that can accept write. These are also known as master nodes. The replication in MongoDB is single master so, only one node can accept write operations at a time.

Secondary nodes are known as slave nodes. These are read only nodes that replicate from the primary.

80. By default, which replica sets are used to write data?

Answer: By default, MongoDB writes data only to the primary replica set.

81. What is CRUD in MongoDB?

Answer: MongoDB supports following CRUD operations:

- Create

- Read
- Update
- Delete

82. In which format MongoDB represents document structure?

Answer: MongoDB uses BSON to represent document structures.

83. What will happen when you remove a document from database in MongoDB? Does MongoDB remove it from disk?

Answer: Yes. If you remove a document from database, MongoDB will remove it from disk too.

84. Why are MongoDB data files large in size?

Answer: MongoDB doesn't follow file system fragmentation and pre allocates data files to reserve space while setting up the server. That's why MongoDB data files are large in size.

85. What is a storage engine in MongoDB?

Answer: A storage engine is the part of a database that is used to manage how data is stored on disk.

For example: one storage engine might offer better performance for read-heavy workloads, and another might support a higher-throughput for write operations.

86. Which are the storage engines used by MongoDB?

Answer: MMAPv1 and WiredTiger are two storage engine used by MongoDB.

87. What is the usage of profiler in MongoDB?

Answer: A database profiler is used to collect data about MongoDB write operations, cursors, database commands on a running mongod instance. You can enable profiling on a per-database or per-instance basis.

The database profiler writes all the data it collects to the system. profile collection, which is a capped collection.

88. Is it possible to configure the cache size for MMAPv1 in MongoDB?

Answer: No. it is not possible to configure the cache size for MMAPv1 because MMAPv1 does not allow configuring the cache size.

89. What is data modeling?

Answer: This term is being used in the context of Mongoose and MongoDB. Data modeling is the procedure of creating the data model for data at hand and after which it can be stored in a database. The data model represents the data object, its relation, and rules which define the connections.

With data modeling, data visualization can be represented while enforcing the data's business rules, compliance, and policy. Data modeling is executed to ensure consistency on convention, values, semantics, security, and data quality.

90. State the meaning of NPM in NodeJS

Answer: NPM can be said as a Node package manager. The main functionalities of NPM are given below.

● NPM works like the command-line utility for installing the packages. NPM carries out the management and dependency version of NodeJS packages.

● NPM can also work as an online repository in the NodeJS packages. It can be present in the .org file.