1. **What is a context API? How does it work?**

Answer:

The Context API is a React structure that allows unique facts to be communicated and assists in the resolution of prop-drilling concerns at various phases of a project.

This is a different approach to "prop drilling," which involves passing props from grandparent to child to parent and so on. Context is also marketed as a more simple and lightweight Redux state management solution.

Users must be familiar with React.createContext(). It will help us connect with a customer and a provider. A provider, as the name implies, is a component that provides the state to its offspring. It will contain the "store" and serve as the parent component for any other components that require it. A component that consumes and uses the state is referred to as a consumer. More details can be found on the React documentation page.

2. **Difference between useEffect and useState?**

Answer:

useState

The useState hook takes one input, the initial state, and returns two values: the current state and a function for updating the state.

useEffect

The useEffect Hook's objective is to eliminate the disadvantages of using class-based components. Setting up listeners, receiving data from the API, and deleting listeners before the component is removed from the DOM are all possible using this hook.

3. **What is JSX? How does it work**

Answer:

JSX is the abbreviation for JavaScript XML. We can write HTML in React due to JSX. JSX simplifies the creation and addition of HTML in React.

Browsers do not support JSX since it is a combination of HTML and JavaScript. As a result, when a file contains JSX, the Babel transpiler turns it to JavaScript objects, resulting in lawful JavaScript. Browsers are able to understand and execute the code as a result.

4. **What is the purpose of a custom hook? How will you create a custom hook? Give us an example.**

Answer:

Custom Hooks are a way to reuse stateful logic, but each time you use one, all state and effects within it are completely isolated.

However, React only provides a few built-in Hooks (such as useReducer, useCallback, useMemo, and useContext). React developers, on the other hand, can construct their own custom hooks by using existing Hooks as a framework.

5. **How would you optimize a react js application**

Answer:

- Keep the component state local when it's necessary.
- To avoid unnecessary re-renders, React components should be remembered.
- Splitting React code with dynamic import ()
- Windowing or list virtualization is utilized in React apps.
- Lazy image loading is possible in React.

6. **How will you send data from a Child Component to the parent component?**

Answer:

The steps for passing data from a child component to a parent component are as follows:

-In the parent component, create a callback function. This callback function will be used to retrieve data from the child component.

-Pass the callback function to the child component as a prop from the parent component.

-The data is supplied to the parent component by the child component, which calls the parent callback method using props.

7. **Is there any reason to return something from a UseEffect hook?**

Answer:

useEffect provides the return function, which is used for cleanup functions.

If I create a subscription for something and then wish to unsubscribe from it, I should put the unsubscription logic in useEffect's "return function" rather than elsewhere, as this could lead to a race condition!

8. **How will you optimize a react application?**

Answer:

I will optimize a react application

When it's necessary, keep the component state local.

To avoid unwanted re-renders, memorize React components.

React code splitting with dynamic import ()

In React, this is known as windowing or list virtualization.

In React, lazy image loading is possible.

9. **What are the different ways to manage state in a React Application**

Answer:

The different ways to manage state in a React Application are -

- Local state
- Global state
- Server state
- URL state

10. **When to use a Class Component over a Function Component?**

Answer:

If the component needs *state or lifecycle methods* then use class component otherwise use function component. *However, from React 16.8 with the addition of Hooks, you could use state , lifecycle methods and other features that were only available in class component right in your function component.* *So, it is always recommended to use Function components, unless you need a React functionality whose Function component equivalent is not present yet, like Error Boundaries *

11. **What is the difference between state and props?**

Answer:

Both *props* and *state* are plain JavaScript objects. While both of them hold information that influences the output of render, they are different in their functionality with respect to component. Props get passed to the component similar to function parameters whereas state is managed within the component similar to variables declared within a function.

12. **What is the difference between createElement and cloneElement?**

Answer:

JSX elements will be transpiled to React.createElement() functions to create React elements which are going to be used for the object representation of UI. Whereas cloneElement is used to clone an element and pass it new props.

13. Is it mandatory to define constructor for React component?

Answer:

No, it is not mandatory. i.e, If you don't initialize state and you don't bind methods, you don't need to implement a constructor for your React component.

14. What are default props?

Answer:

The defaultProps are defined as a property on the component class to set the default props for the class. This is used for undefined props, but not for null props.

For example, let us create color default prop for the button component,

```
class MyButton extends React.Component {
 // ...
}

MyButton.defaultProps = {
 color: 'red'
};
```

If props.color is not provided then it will set the default value to 'red'. i.e, Whenever you try to access the color prop it uses default value

```
render() {
  return <MyButton /> ; // props.color will be set to red
}
```

Note: If you provide null value then it remains null value.

15. How to remove attribute from MongoDB Object?

Answer:

$unset

The $unset operator deletes a particular field. If the field does not exist, then $unset does nothing. When used with $ to match an array element, $unset replaces the

matching element with null rather than removing the matching element from the array. This behavior keeps consistent the array size and element positions.

syntax:

{ $unset: { <field1>: "", ... } }

16. What is "Namespace" in MongoDB?

Answer:

MongoDB stores BSON (Binary Interchange and Structure Object Notation) objects in the collection. The concatenation of the collection name and database name is called a namespace.

17. What is Replication in Mongodb?

Answer:

Replication exists primarily to offer data redundancy and high availability. It maintain the durability of data by keeping multiple copies or replicas of that data on physically isolated servers. Replication allows to increase data availability by creating multiple copies of data across servers. This is especially useful if a server crashes or hardware failure.

With MongoDB, replication is achieved through a Replica Set. Writer operations are sent to the primary server (node), which applies the operations across secondary servers, replicating the data. If the primary server fails (through a crash or system failure), one of the secondary servers takes over and becomes the new primary node via election. If that server comes back online, it becomes a secondary once it fully recovers, aiding the new primary node.

18. When should we embed one document within another in MongoDB?

Answer:

We should consider embedding documents for:

contains relationships between entities

One-to-many relationships

Performance reasons

19. How is data stored in MongoDB?

Answer:

In MongoDB, Data is stored in BSON documents (short for Binary JSON). These documents are stored in MongoDB in JSON (JavaScript Object Notation) format. JSON documents support embedded fields, so related data and lists of data can be stored with the document instead of an external table. Documents contain one or more fields, and each field contains a value of a specific data type, including arrays, binary data and sub-documents. Documents that tend to share a similar structure are organized as collections.

JSON is formatted as name/value pairs. In JSON documents, field names and values are separated by a colon, field name and value pairs are separated by commas, and sets of fields are encapsulated in "curly braces" ({}).

Example:

```
{
  "name": "notebook",
  "qty": 50,
  "rating": [ { "score": 8 }, { "score": 9 } ],
  "size": { "height": 11, "width": 8.5, "unit": "in" },
  "status": "A",
  "tags": [ "college-ruled", "perforated"]
}
```

20. What are the differences between MongoDB and SQL-SERVER?

Answer:

-The MongoDB store the data in documents with JSON format but SQL store the data in Table format.

-The MongoDB provides high performance, high availability, easy scalability etc. rather than SQL Server.

-In the MongoDB, we can change the structure simply by adding, removing column from the existing documents.

21. Explain limitations of MongoDB Transactions?

Answer:

MongoDB transactions can exist only for relatively short time periods. By default, a transaction must span no more than one minute of clock time. This limitation results from the underlying MongoDB implementation. MongoDB uses MVCC, but unlike databases such as Oracle, the “older” versions of data are kept only in memory.

-You cannot create or drop a collection inside a transaction.

-Transactions cannot make writes to a capped collection

-Transactions take plenty of time to execute and somehow they can slow the performance of the database.

-Transaction size is limited to 16MB requiring one to split any that tends to exceed this size into smaller transactions.

-Subjecting a large number of documents to a transaction may exert excessive pressure on the WiredTiger engine and since it relies on the snapshot capability, there will be a retention of large unflushed operations in memory. This renders some performance cost on the database.

22. Does MongoDB pushes the writes to disk immediately or lazily?

Answer:

MongoDB pushes the data to disk lazily. It updates the immediately written to the journal but writing the data from journal to disk happens lazily.

23. How to perform a delete operation in MongoDB?

Answer:

MongoDB's db.collection.deleteMany() and db.collection.deleteOne() method is used to delete documents from the collection. Delete operations do not drop indexes, even if deleting all documents from a collection. All write operations in MongoDB are atomic on the level of a single document.

Example:

```
// Create Inventory Collection
db.inventory.insertMany( [
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "P" },
```

```
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" },
] );
// Delete Commands
db.inventory.deleteMany({}) // Delete All Documents
db.inventory.deleteMany({ status : "A" }) // Delete All Documents that Match a Condition
db.inventory.deleteOne( { status: "D" } ) // Delete Only One Document that Matches a Condition
```

24. If you remove a document from the database, does MongoDB remove it from disk?

Answer:

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

25. What is a covered query in MongoDB?

Answer:

The MongoDB covered query is one which uses an index and does not have to examine any documents. An index will cover a query if it satisfies the following conditions:

All fields in a query are part of an index.

All fields returned in the results are of the same index.

No fields in the query are equal to null.

26. Why MongoDB is not preferred over a 32-bit system?

Answer:

When running a 32-bit system build of MongoDB, the total storage size for the server, including data and indexes, is 2 gigabytes. The reason for this is that the MongoDB storage engine uses memory-mapped files for performance.

If you are running a 64-bit build of MongoDB, there is virtually no limit to storage size.

27. Can one MongoDB operation lock more than one database?

Answer:

Yes. Operations like db.copyDatabase(), db.repairDatabase(), etc. can lock more than one databases involved.

28. What is Sharding in MongoDB?

Answer:

Sharding is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

Database systems with large data sets or high throughput applications can challenge the capacity of a single server. For example, high query rates can exhaust the CPU capacity of the server. Working set sizes larger than the system's RAM stress the I/O capacity of disk drives. There are two methods for addressing system growth: vertical and horizontal scaling.

- Vertical Scaling

Vertical Scaling involves increasing the capacity of a single server, such as using a more powerful CPU, adding more RAM, or increasing the amount of storage space.

- Horizontal Scaling

Horizontal Scaling involves dividing the system dataset and load over multiple servers, adding additional servers to increase capacity as required. While the overall speed or capacity of a single machine may not be high, each machine handles a subset of the overall workload, potentially providing better efficiency than a single high-speed high-capacity server.

29. What is Aggregation in MongoDB?

Answer:

Aggregation in MongoDB is an operation used to process the data that returns the computed results. Aggregation basically groups the data from multiple documents and operates in many ways on those grouped data in order to return one combined result.

Aggregate function groups the records in a collection, and can be used to provide total number(sum), average, minimum, maximum etc out of the group selected. In order to perform the aggregate function in MongoDB, aggregate () is the function to be used.

Syntax

db.collection_name.aggregate(aggregate_operation)

MongoDB provides three ways to perform aggregation:

-the aggregation pipeline,

-the map-reduce function,

-and single purpose aggregation methods and commands.

30. Why are MongoDB data files large in size?

Answer:

MongoDB preallocate data files to reserve space and avoid file system fragmentation when you set up the server.

31. How can you isolate your cursors from intervening with the write operations?

Answer:

As the cursor is not isolated during its lifetime, thus intervening write operations on a document may result in a cursor that returns a document more than once. The snapshot() method can be used on a cursor to isolate the operation for a very specific case. snapshot() traverses the index on the _id field and guarantees that the query will return each document no more than once.

Restriction:

We cannot use snapshot() with sharded collections.

We cannot use snapshot() with sort() or hint() cursor methods.

32. At what interval does MongoDB write updates to the disk?

Answer:

By default configuration, MongoDB writes updates to the disk every 60 seconds. However, this is configurable with the commitIntervalMs and syncPeriodSecs options.

33. What happens if an index does not fit into RAM?

Answer:

If the indexes does not fit into RAM, MongoDB reads data from disk which is relatively very much slower than reading from RAM.

Indexes do not have to fit entirely into RAM in all cases. If the value of the indexed field increments with every insert, and most queries select recently added documents; then MongoDB only needs to keep the parts of the index that hold the most recent or "right-most" values in RAM. This allows for efficient index use for read and write operations and minimizes the amount of RAM required to support the index.

Example: To check the size of indexes

> db.collection.totalIndexSize()

// Output (in bytes)

4294976499

34. How does Journaling work in MongoDB?

Answer:

Mongod primarily hosts the write operations in memory in a shared view. It is called shared because it has memory mapping in the actual disc. In this process, a write operation occurs in mongod, which then creates changes in the private view. The first block is memory and the second block is "my disc". After a specified interval, which is called a "journal commit interval", the private view writes those operations in the journal directory (residing in the disc).

Once the journal commit happens, mongod pushes data into a shared view. As part of the process, it gets written to the actual data directory from the shared view (as this process happens in background). The basic advantage is, we have a reduced cycle from 60 seconds to 200 milliseconds.

In a scenario where an abruption occurs at any point of time or flash disc remains unavailable for the last 59 seconds , then when the next time mongod starts, it basically replays all write operation logs and writes into the actual data directory.

35. **Is MongoDB schema-less?**

Answer:

As a NoSQL database, MongoDB is considered schemaless because it does not require a rigid, pre-defined schema like a relational database. The database management system (DBMS) enforces a partial schema as data is written, explicitly listing collections and indexes.

MongoDB is a document based database, which does not use the concept of tables and columns, instead of which it uses the concept of documents and collections. All the referential data with respect to different modules will be stored as one collection. Moreover, the BSON data structure used by MongoDB can easily have varying sets of data and fields with different types.

When we say schemaless, we actually mean dynamically typed schema, as opposed to statically typed schemas as available in RDBMS(SQL) databases. JSON is a

completely schema free data structure, as opposed to XML which allows you to specify XSD if you need.

36. How to condense large volumes of data in Mongodb?

Answer:

compact

Rewrites and defragments all data and indexes in a collection. On WiredTiger databases, this command will release unneeded disk space to the operating system. This command will perform a compaction "in-line".

MongoDB compresses the files by:

copying the files to a new location

looping through the documents and re-ordering / re-solving them

replacing the original files with the new files

Syntax

{ compact: <collection name> }

37. What is splitting in MongoDB?

Answer:

Splitting is a process that keeps chunks from growing too large. When a chunk grows beyond a specified chunk size, or if the number of documents in the chunk exceeds Maximum Number of Documents Per Chunk to Migrate, MongoDB splits the chunk based on the shard key values the chunk represents.

38. Explain what is horizontal scalability in mongodb?

Answer:

Horizontal Scaling involves dividing the system dataset and load over multiple servers, adding additional servers to increase capacity as required. While the overall speed or capacity of a single machine may not be high, each machine handles a subset of the overall workload, potentially providing better efficiency than a single high-speed high-capacity server. Expanding the capacity of the deployment only requires adding additional servers as needed, which can be a lower overall cost than high-end hardware for a single machine. The trade off is increased complexity in infrastructure and maintenance for the deployment.

39. sql vs nosql মধ্যে পার্থক্য কি**?** কোনটা বেশি ব্যবহার হয়

Answer:

- Sql means Structure Query Language.
- On the other hand, nonsql Not only Structure Query Language.
- In SQL databases, Shema is predefined.
- NoSQL databases use dynamic schema for unstructured data.

Sql is very useful.

40. রিয়েক্ট এর সাথ **node, Mongodb** কেন ইউস করা হয়ছে **MySql** কেন নয়**?**

Answer:

- Architecture in the Component Style
- Instant updates without reloading the page
- SEO Friendly Features
- The creation of dynamic web apps is simple.
- Uses third party components
- Easy to learn
- Transitioning to React Native is simple.

41. Database design, database schema design বলতে কি বুজো**?**

Answer:

Database design refers to a set of procedures that make it easier to plan, create, implement, and maintain enterprise data management systems. A well-designed database is simple to manage, increases data consistency, and saves money on disk storage space.

The database schema design splits data into numerous entities, explains how to connect the organized entities, and sets data constraints. Designers build database schemas to give other database users, such as programmers and analysts, a logical understanding of the data.

42. সার্ভার সাইট ক্রাস করলে কি করবেন**?**

Answer:

Step 1: Determine the source of the problem.

Step 2: Resolve the problem

43. API কিভাবে কাজ করে

Answer:

APIs work by transferring data and information between applications, systems, and devices, allowing them to communicate with one another.

44. What is CRUD

Answer: CRUD means Create, Read, Update, Delete operations.

45. Get vs post

Answer:

-We can't send big amounts of data using the GET technique since the request parameter is tied to the URL.

-The POST method can handle enormous amounts of data because the request parameter is appended to the body.

46. PUT and Patch এর মধ্যে পার্থক্য কি**?.**

Answer:

- PUT is most commonly used for update capabilities, with the request body holding the newly-updated representation of the original resource.
- According to the PATCH request, we would only submit the data that has to be modified, without updating or affecting other portions of the data. For example, if we simply need to update the first name, we pass only the first name.
- PUT is used to see if a resource already exists and then update it; otherwise, it is used to create a new resource.
- PATCH is used to update a resource.

47. How will you secure an API

Answer:

Transport Layer Security should be used by all Web APIs. TLS encrypts all communications in transit, assuring the security of data provided through the API. TLS and its forerunner, SSL, are sometimes used interchangeably. TLS is enabled when a website's URL begins with https:// rather than http://.

I will secure an API-

-Vulnerabilities need to be identified.

-Use OAuth

-Encrypt Data

-Use Throttling and Rate Limiting

-Test Your APIs with Dynamic Application Security Testing -Use a Service Mesh

48. **Mongoose** কি**?** কীভাবে কাজ করে**?** এটা নিয়ে কাজ করেছো কিনা

Answer:

Mongoose is a way to connect to the MongoDB database. It simplifies MongoDB validation and querying, allowing developers to work more quickly.

Mongoose manages data associations, does schema validation, and is used to translate between objects in code and their MongoDB representations. MongoDB is a NoSQL document database with no schema.

For the time being, I'm working in Mongoose, but I'll be working in the near future.

49. What is Multi-threading?

Answer:

Multi-threading is a process of improving the performance of CPU. Generally, it is the ability of a program to get managed by more than one user at a time or manage multiple requests by the same user. Multi-threading is done by executing multiple processes that can be supported by the operating system.

50. What is the main difference between GraphQL and REST?

Answer:

This is a moderately difficult question but a good developer would be able to get through with this in ease. An important difference between GraphQL and REST is that GraphQL doesn’t deal with dedicated resources. Everything referred to as a graph is connected and can be queried to app needs.

51. List some common ways to reduce the load time of a web application?

Answer:

There are quite a lot of ways to reduce the loading time of a web application like enabling browser caching, optimizing images, minifying resources, minimizing HTTP requests and reducing redirects.