INTERVIEW QUESTIONS ON MERN STACK DEVELOPMENT

How does React work?

React creates a virtual DOM. When state changes in a component it firstly runs a "diffing" algorithm, which identifies what has changed in the virtual DOM. The second step is reconciliation, where it updates the DOM with the results of diff.

What are the advantages of ReactJS?

Below are the advantages of ReactJS:

1. Increases the application's performance with Virtual DOM
2. JSX makes code is easy to read and write
3. It renders both on client and server side
4. Easy to integrate with other frameworks (Angular, BackboneJS) since it is only a view library
5. Easy to write UI Test cases and integration with tools such as JEST.

What is props in React?

Props are inputs to a React component. They are single values or objects containing a set of values that are passed to React Components on creation using a naming convention similar to HTML-tag attributes. i.e, They are data passed down from a parent component to a child component.

The primary purpose of props in React is to provide following component functionality:

1. Pass custom data to your React component.
2. Trigger state changes.
3. Use via this.props.reactProp inside component's render() method.

For example, let us create an element with reactProp property,

```
<Element reactProp = "1" />
```

This reactProp (or whatever you came up with) name then becomes a property attached to React's native props object which originally already exists on all components created using React library.

```
props.reactProp;
```

Which are the most important features of MongoDB?

- Flexible data model in form of documents
- Agile and highly scalable database
- Faster than traditional databases
- Expressive query language

How is React different from AngularJS (1.x)?

For example, AngularJS (1.x) approaches building an application by extending HTML markup and injecting various constructs (e.g. Directives, Controllers, Services) at runtime. As a result, AngularJS is very opinionated about the greater architecture of your application — these abstractions are certainly useful in some cases, but they come at the cost of flexibility.

By contrast, React focuses exclusively on the creation of components, and has few (if any) opinions about an application's architecture. This allows a developer an incredible amount of flexibility in choosing the architecture they deem "best" — though it also places the responsibility of choosing (or building) those parts on the developer.

What Is Replication In MongoDB?

Replication is the process of synchronizing data across multiple servers. Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions.

What are Higher-Order components?

A higher-order component **(HOC)** is a function that takes a component and returns a new component. Basically, it's a pattern that is derived from React's compositional nature We call them as **"pure' components"** because they can accept any dynamically provided child component but they won't modify or copy any behavior from their input components.

```
const EnhancedComponent = higherOrderComponent(WrappedComponent);
```

HOC can be used for many use cases as below,

1. Code reuse, logic and bootstrap abstraction
2. Render High jacking
3. State abstraction and manipulation
4. Props manipulation

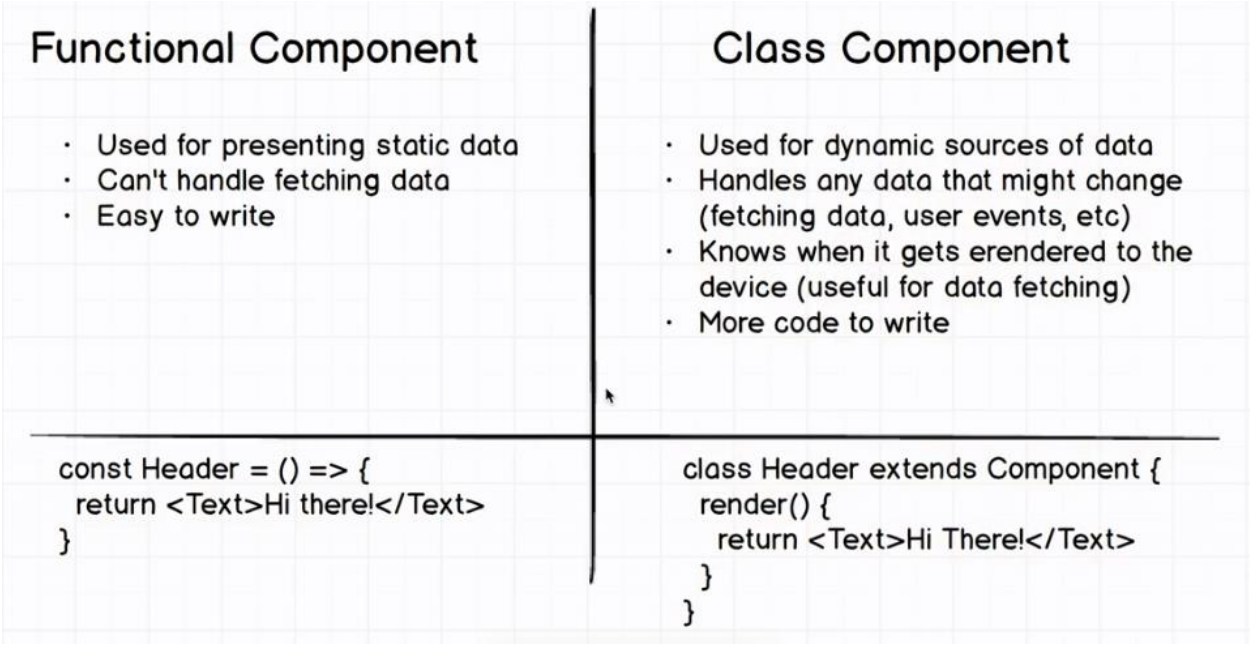
What are the differences between a class component and functional component?

Class Components

- Class-based Components uses ES6 class syntax. It can make use of the lifecycle methods.
- Class components extend from React.Component.
- In here you have to use this keyword to access the props and functions that you declare inside the class components.

Functional Components

- Functional Components are simpler comparing to class-based functions.
- Functional Components mainly focuses on the UI of the application, not on the behavior.
- To be more precise these are basically render function in the class component.
- Functional Components can have state and mimic lifecycle events using Reach Hooks



What are the key features of Node.js?

Let's look at some of the key features of Node.js.

- **Asynchronous event driven IO helps concurrent request handling –** All APIs of Node.js are asynchronous. This feature means that if a Node receives a request for some Input/Output operation, it will execute that operation in the background and continue with the processing of other requests. Thus it will not wait for the response from the previous requests.
- **Fast in Code execution –** Node.js uses the V8 JavaScript Runtime engine, the one which is used by Google Chrome. Node has a wrapper over the JavaScript engine which makes the runtime engine much faster and hence processing of requests within Node.js also become faster.
- **Single Threaded but Highly Scalable –** Node.js uses a single thread model for event looping. The response from these events may or may not reach the server immediately. However, this does not block other operations. Thus making Node.js highly scalable. Traditional servers create limited threads to handle requests while Node.js creates a single thread that provides service to much larger numbers of such requests.

- **Node.js library uses JavaScript –** This is another important aspect of Node.js from the developer's point of view. The majority of developers are already well-versed in JavaScript. Hence, development in Node.js becomes easier for a developer who knows JavaScript.
- **There is an Active and vibrant community for the Node.js framework –** The active community always keeps the framework updated with the latest trends in the web development.
- **No Buffering –** Node.js applications never buffer any data. They simply output the data in chunks.

What are the limitations of React?

Below are the list of limitations:

1. React is just a view library, not a full-blown framework
2. There is a learning curve for beginners who are new to web development.
3. Integrating React.js into a traditional MVC framework requires some additional configuration
4. The code complexity increases with inline templating and JSX.
5. Too many smaller components leading to over-engineering or boilerplate

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 is Callback Hell?

The asynchronous function requires callbacks as a return parameter. When multiple asynchronous functions are chained together then callback hell situation comes up.

What is Reconciliation?

When a component's props or state change, React decides whether an actual DOM update is necessary by comparing the newly returned element with the previously rendered one. When they are not equal, React will update the DOM. This process is called **reconciliation**.

What is the difference between returning a callback and just calling a callback?

```
return callback();

//some more lines of code; -  won't be executed


callback();
```

```
//some more lines of code; - will be executed
```

Of course returning will help the context calling async function get the value returned by callback.

```
function do2(callback) {
    log.trace('Execute function: do2');
    return callback('do2 callback param');
}

var do2Result = do2((param) => {
    log.trace(`print ${param}`);
    return `return from callback(${param})`; // we could use that return
});

log.trace(`print ${do2Result}`);
```

Output:

```
C:\Work\Node>node --use-strict main.js
[0] Execute function: do2
[0] print do2 callback param
[0] print return from callback(do2 callback param)
```

When should we embed one document within another in MongoDB?

You should consider embedding documents for:

- contains relationships between entities
- One-to-many relationships
- Performance reasons

Does Mongodb support Foreign Key constraints?

No. MongoDB does not support such relationships. The database does not apply any constraints to the system (i.e.: foreign key constraints), so there are no "cascading deletes" or "cascading updates". Basically, in a NoSQL database it is up to you to decide how to organise the data and its relations if there are any.

Explain advantages of BSON over JSON in MongoDB?

- **BSON** is designed to be efficient in space, but in some cases is not much more efficient than JSON. In some cases BSON uses even more space than JSON. The reason for this is another of the BSON design goals: traversability. BSON adds some "extra" information to documents, like length of strings and subobjects. This makes traversal faster.
- BSON is also designed to be fast to encode and decode. For example, integers are stored as 32 (or 64) bit integers, so they don't need to be parsed to and from text. This uses more space than JSON for small integers, but is much faster to parse.
- In addition to compactness, BSON adds additional data types unavailable in JSON, notably the BinData and Date data types.

Given the code defined above, can you identify two problems?

Answer

Take a look at the code below:

```
class MyComponent extends React.Component {
 constructor(props) {
  // set the default internal state
  this.state = {
    clicks: 0
  };
 }

 componentDidMount() {
  this.refs.myComponentDiv.addEventListener('click', this.clickHandler);
 }

 componentWillUnmount() {
  this.refs.myComponentDiv.removeEventListener('click', this.clickHandler);
 }

 clickHandler() {
  this.setState({
```

```
    clicks: this.clicks + 1
  });
 }

 render() {
  let children = this.props.children;

  return (
   <div className="my-component" ref="myComponentDiv">
   <h2>My Component ({this.state.clicks} clicks})</h2>
   <h3>{this.props.headerText}</h3>
  {children}
  </div>
  );
 }
}
```

Given the code defined above, can you identify two problems?

Answer: 1. The constructor does not pass its props to the super class. It should include the following line:

```
constructor(props) {
 super(props);
 // ...
}
```

1. The event listener (when assigned via addEventListener()) is not properly scoped because ES2015 doesn't provide autobinding. Therefore the developer can re-assign clickHandler in the constructor to include the correct binding to this:

```
constructor(props) {
 super(props);
 this.clickHandler = this.clickHandler.bind(this);
 // ...
}
```

How Node prevents blocking code?

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

How can you achieve transaction and locking in MongoDB?

To achieve concepts of transaction and locking in MongoDB, we can use the nesting of documents, also called embedded (or sub) documents. MongoDB supports atomic operations within a single document.

How does Node.js handle child threads?

Node.js, in its essence, is a single thread process. It does not expose child threads and thread management methods to the developer. Technically, Node.js does spawn child threads for certain tasks such as asynchronous I/O, but these run behind the scenes and do not execute any application JavaScript code, nor block the main event loop.

If threading support is desired in a Node.js application, there are tools available to enable it, such as the ChildProcess module.

How to avoid Callback Hell in Node.js?

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. Sometimes, it could lead to complex and unreadable code. More the no. of callbacks, longer the chain of returning callbacks would be.

There are four solutions which can address the callback hell problem:

- **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.
- **Use async/await mechanism** - Async /await is another alternative for consuming promises, and it was implemented in ES8, or ES2017. Async/await is a new way of writing promises that are based on asynchronous code but make asynchronous code look and behave more like synchronous code.
- **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.
- **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.

```
function* HelloGen() {
  yield 100;
  yield 400;
}

var gen = HelloGen();

console.log(gen.next()); // {value: 100, done: false}
console.log(gen.next()); // {value: 400, done: false}
console.log(gen.next()); // {value: undefined, done: true}
```

How to query MongoDB with like?

I want to query something as SQL's like query:

```
select *
from users
where name like '%m%'
```

How to do the same in MongoDB?

```
db.users.find({"name": /.*m.*/})
// or
db.users.find({"name": /m/})
```

You're looking for something that contains "m" somewhere (SQL's '%' operator is equivalent to Regexp's '.*'), not something that has "m" anchored to the beginning of the string.

If Node.js is single threaded then how it handles concurrency?

Node provides a single thread to programmers so that code can be written easily and without bottleneck. Node internally uses multiple POSIX threads for various I/O operations such as File, DNS, Network calls etc.

When Node gets I/O request it creates or uses a thread to perform that I/O operation and once the operation is done, it pushes the result to the **event queue**. On each such

event, **event loop** runs and checks the queue and if the execution stack of Node is empty then it adds the queue result to execution stack.

This is how Node manages concurrency.

Rewrite promise-based Node.js applications to Async/Await

Problem

Rewrite this code to Async/Await:

```
function asyncTask() {
  return functionA()
    .then((valueA) => functionB(valueA))
    .then((valueB) => functionC(valueB))
    .then((valueC) => functionD(valueC))
    .catch((err) => logger.error(err))
}
```

Answer

```
async function asyncTask() {
  try {
    const valueA = await functionA()
    const valueB = await functionB(valueA)
    const valueC = await functionC(valueB)
    return await functionD(valueC)
  } catch (err) {
    logger.error(err)
  }
}
```

What are Pure Components?

PureComponent is exactly the same as Component except that it handles the shouldComponentUpdate method for you. When props or state changes, PureComponent will do a shallow comparison on both props and state. Component, on the other hand, won't compare current props and state to next out of the box. Thus, the component will re-render by default whenever shouldComponentUpdate is called.

What are React Hooks?

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class. With Hooks, you can extract stateful logic from a component so it can be tested independently and reused. Hooks allow you to reuse stateful logic without changing your component hierarchy. This makes it easy to share Hooks among many components or with the community.

What are advantages of using React Hooks?

Primarily, hooks in general enable the extraction and reuse of stateful logic that is common across multiple components without the burden of higher order components or render props. Hooks allow to easily manipulate the state of our functional component without needing to convert them into class components.

Hooks don't work inside classes (because they let you use React without classes). By using them, we can totally avoid using lifecycle methods, such as componentDidMount, componentDidUpdate, componentWillUnmount. Instead, we will use built-in hooks like useEffect .

What is ReactDOM?

AnswerIt's a top-level React API to render a React element into the DOM, via the ReactDOM.render method.

What is Aggregation in MongoDB?

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation:

- the aggregation pipeline,
- the map-reduce function,
- and single purpose aggregation methods and commands.

What is Sharding in MongoDB?

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

What is Stream and what are types of Streams available in Node.js?

Streams are a collection of data that might not be available all at once and don't have to fit in memory. Streams provide chunks of data in a continuous manner. It is useful to read a large set of data and process it.

There is four fundamental type of streams:

- Readable.
- Writeable.

- Duplex.
- Transform.

Readable streams as the name suggest used in reading a large chunk of data from a source. Writable streams are used in writing a large chunk of data to the destination.

Duplex streams are both readable and writable (Eg socket). Transform stream is the duplex stream which is used in modifying the data (eg zip creation).

What is prop drilling and how can you avoid it?

When building a React application, there is often the need for a deeply nested component to use data provided by another component that is much higher in the hierarchy. The simplest approach is to simply pass a prop from each component to the next in the hierarchy from the source component to the deeply nested component. This is called **prop drilling**.

The primary disadvantage of prop drilling is that components that should not otherwise be aware of the data become unnecessarily complicated and are harder to maintain.

To avoid prop drilling, a common approach is to use React context. This allows a Provider component that supplies data to be defined, and allows nested components to consume context data via either a Consumer component or a useContext hook.

What is Key and benefit of using it in lists?

A **key** is a special string attribute you need to include when creating lists of elements. Keys help React identify which items have changed, are added, or are removed.

For example, most often we use IDs from your data as keys

```
const todoItems = todos.map((todo) =>
   <li key={todo.id}>
   {todo.text}
   </li>
);
```

When you don’t have stable IDs for rendered items, you may use the item index as a key as a last resort:

```
const todoItems = todos.map((todo, index) =>
   <li key={index}>
      {todo.text}
   </li>
);
```

Note:

1. We don’t recommend using indexes for keys if the order of items may change. This can negatively impact performance and may cause issues with component state
2. If you extract list item as separate component then apply keys on list component instead li tag.

There will be a warning in the console if the key is not present on list items.

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.

What is the alternative of binding this in the constructor?

You can use property initializers to correctly bind callbacks. This is enabled by default in create react app. You can use an arrow function in the callback. The problem here is that a new callback is created each time the component renders.

What is the difference between ShadowDOM and VirtualDOM?

Virtual DOM

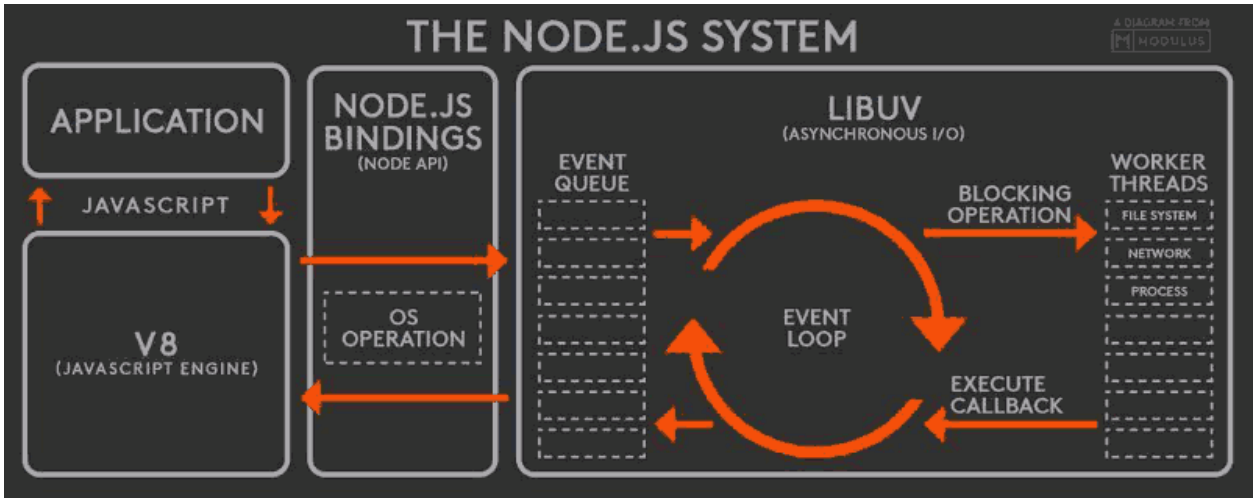
Virtual DOM is about avoiding unnecessary changes to the DOM, which are expensive performance-wise, because changes to the DOM usually cause re-rendering of the page. Virtual DOM also allows to collect several changes to be applied at once, so not every single change causes a re-render, but instead re-rendering only happens once after a set of changes was applied to the DOM.

Shadow DOM

Shadow dom is mostly about encapsulation of the implementation. A single custom element can implement more-or-less complex logic combined with more-or-less complex DOM. An entire web application of arbitrary complexity can be added to a page by an import and <body><my-app></my-app> but also simpler reusable and composable components can be implemented as custom elements where the internal representation is hidden in the shadow DOM like <date-picker></date-picker>.

What's the Event Loop?

The event loop is what allows Node.js to perform non-blocking I/O operations — despite the fact that JavaScript is single-threaded — by offloading operations to the system kernel whenever possible.



Every I/O requires a callback - once they are done they are pushed onto the event loop for execution. Since most modern kernels are multi-threaded, they can handle multiple operations executing in the background. When one of these operations completes, the kernel tells Node.js so that the appropriate callback may be added to the poll queue to eventually be executed.

What's the difference between useRef and createRef?

The difference is:

- createRef will always create a new ref. In a class-based component, you would typically put the ref in an instance property during construction (e.g. this.input = createRef()). You don't have this option in a function component.
- useRef takes care of returning the same ref each time as on the initial rendering.

What's the difference between a "smart" component and a "dumb" component?

Smart components manage their state or in a Redux environment are connected to the Redux store.

Dumb components are driven completely by their props passed in from their parent and maintain no state of their own.

How does React work

React creates a virtual DOM. When state changes in a component it firstly runs a "diffing" algorithm, which identifies what has changed in the virtual DOM. The second step is reconciliation, where it updates the DOM with the results of diff.

Q.2What is props in React

Props are inputs to a React component. They are single values or objects containing a set of values that are passed to React Components on creation using a naming convention similar to HTML-tag attributes. i.e, They are data passed down from a parent component to a child component. The primary purpose of props in React is to provide following component functionality: Pass custom data to your React

component, Trigger state changes and Use via this.props.reactProp inside component's render() method.

Q.3What Is Replication In MongoDB

Replication is the process of synchronizing data across multiple servers. Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions.

Q.4What are Higher-Order components

A higher-order component (HOC) is a function that takes a component and returns a new component. Basically, it's a pattern that is derived from React's compositional nature We call them as "pure' components" because they can accept any dynamically provided child component but they won't modify or copy any behavior from their input components.

Q.5What 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.

Q.6What is Callback Hell

The asynchronous function requires callbacks as a return parameter. When multiple asynchronous functions are chained together then callback hell situation comes up.

Q.7What is Reconciliation

When a component's props or state change, React decides whether an actual DOM update is necessary by comparing the newly returned element with the previously rendered one. When they are not equal, React will update the DOM. This process is called reconciliation.

Q.8 Does Mongodb Support Foreign Key Constraints

No. MongoDB does not support such relationships. The database does not apply any constraints to the system (i.e.: foreign key constraints), so there are no "cascading deletes" or "cascading updates". Basically, in a NoSQL database it is up to you to decide how to organise the data and its relations if there are any.

Q.9How Node prevents blocking code

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

Q.10How can you achieve transaction and locking in MongoDB

To achieve concepts of transaction and locking in MongoDB, we can use the nesting of documents, also called embedded (or sub) documents. MongoDB supports atomic operations within a single document.

Q.11How does Node.js handle child threads

Node.js, in its essence, is a single thread process. It does not expose child threads and thread management methods to the developer. Technically, Node.js does spawn child threads for certain tasks such as asynchronous I/O, but these run behind the scenes and do not execute any application JavaScript code, nor block the main event loop. If threading support is desired in a Node.js application, there are tools available to enable it, such as the ChildProcess module.

Q.12How to avoid Callback Hell in Node.js

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. Sometimes, it could lead to complex and unreadable code. More the no. of callbacks, longer the chain of returning callbacks would be. There are four solutions which can address the callback hell problem: Make your program modular, Use async/await mechanism, Use promises mechanism and Use generators

Q.13If Node.js is single threaded then how it handles concurrency

Node provides a single thread to programmers so that code can be written easily and without bottleneck. Node internally uses multiple POSIX threads for various I/O operations such as File, DNS, Network calls etc. When Node gets I/O request it creates or uses a thread to perform that I/O operation and once the operation is done, it pushes the result to the event queue. On each such event, event loop runs and checks the queue and if the execution stack of Node is empty then it adds the queue result to execution stack.

Q.14What are Pure Components

PureComponent is exactly the same as Component except that it handles the shouldComponentUpdate method for you. When props or state changes, PureComponent will do a shallow comparison on both props and state. Component, on the other hand, won't compare current props and state to next out of the box. Thus, the component will re-render by default whenever shouldComponentUpdate is called.

Q.15What are React Hooks

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class. With Hooks, you can extract stateful logic from a component so it can be tested independently and reused. Hooks allow you to reuse

stateful logic without changing your component hierarchy. This makes it easy to share Hooks among many components or with the community.

Q.16What is Aggregation in MongoDB

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function and single purpose aggregation methods and commands.

Q.17What is JSX

JSX is a syntax extension to JavaScript and comes with the full power of JavaScript. JSX produces React elements. You can embed any JavaScript expression in JSX by wrapping it in curly braces. After compilation, JSX expressions become regular JavaScript objects. This means that you can use JSX inside of if statements and for loops, assign it to variables, accept it as arguments, and return it from functions:

Q.18What is ReactDOM

It's a top-level React API to render a React element into the DOM, via the ReactDOM.render method.

Q.19What is Sharding in MongoDB

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

Q.20What is Stream and what are types of Streams available in Node.js

Streams are a collection of data that might not be available all at once and don't have to fit in memory. Streams provide chunks of data in a continuous manner. It is useful to read a large set of data and process it. There is four fundamental type of streams: Readable, Writeable, Duplex and Transform

Q.21What is prop drilling

When building a React application, there is often the need for a deeply nested component to use data provided by another component that is much higher in the hierarchy. The simplest approach is to simply pass a prop from each component to the next in the hierarchy from the source component to the deeply nested component. This is called prop drilling.

Q.22What is Key

A key is a special string attribute you need to include when creating lists of elements. Keys help React identify which items have changed, are added, or are removed.

Q.23What 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.

Q.24What is the difference between ShadowDOM and VirtualDOM

Virtual DOM is about avoiding unnecessary changes to the DOM, which are expensive performance-wise, because changes to the DOM usually cause re-rendering of the page. Virtual DOM also allows to collect several changes to be applied at once, so not every single change causes a re-render, but instead re-rendering only happens once after a set of changes was applied to the DOM. Shadow DOM is mostly about encapsulation of the implementation. A single custom element can implement more-or-less complex logic combined with more-or-less complex DOM. An entire web application of arbitrary complexity can be added to a page by an import and but also simpler reusable and composable components can be implemented as custom elements where the internal representation is hidden in the shadow DOM like .

Q.25What's the Event Loop

The event loop is what allows Node.js to perform non-blocking I/O operations — despite the fact that JavaScript is single-threaded — by offloading operations to the system kernel whenever possible. Every I/O requires a callback - once they are done they are pushed onto the event loop for execution. Since most modern kernels are multi-threaded, they can handle multiple operations executing in the background. When one of these operations completes, the kernel tells Node.js so that the appropriate callback may be added to the poll queue to eventually be executed.

Q.26What's the difference between a "smart" component and a "dumb" component

Smart components manage their state or in a Redux environment are connected to the Redux store. Dumb components are driven completely by their props passed in from their parent and maintain no state of their own.

Q.27What is Mongoose

Mongoose is an Object Document Mapper (ODM), which means that by using Mongoose, you can define objects with a strongly-typed schema that can be further mapped to a MongoDB document. It offers a schema-based solution for modeling application data. Mongoose comes with built-in typecasting, validation, query building, business logic hooks, and many more out-of-the-box features.

Q.28What is REPL In Node.Js

REPL or “Read Eval Print Loop” is a simple program that can accept commands, evaluate them, and prints the results. What REPL does is to create an environment that is similar to a Unix/Linux shell or a Window console, wherein you can enter command and system, and it will respond with the output. Here are the functions that REPL performs: READ (This reads the input provided by the user, parses it into

JavaScript data structure, and stores it in the memory.), EVAL (This executes the data structure), PRINT (This prints the outcome generated after evaluating the command.) and LOOP (This loops the above command until the user presses Ctrl+C twice.)

Q.29How to check if an object is an array or not in JavaScript

The best way to find whether an object is instance of a particular class or not using toString method from Object.prototype

Q.30List down the two arguments that async.queue takes as input in Node.js

Task Function and Concurrency Value

Q.31What is the purpose of module.exports in Node.js

This is used to expose functions of a particular module or file to be used elsewhere in the project. This can be used to encapsulate all similar functions in a file which further improves the project structure.

Q.32What is node.js streams

Streams are instances of EventEmitter which can be used to work with streaming data in Node.js. They can be used for handling and manipulating streaming large files(videos, mp3, etc) over the network. They use buffers as their temporary storage.

Q.33What are node.js buffers

In general, buffers is a temporary memory that is mainly used by stream to hold on to some data until consumed. Buffers are introduced with additional use cases than JavaScript's Unit8Array and are mainly used to represent a fixed-length sequence of bytes. This also supports legacy encodings like ASCII, utf-8, etc. It is a fixed(non-resizable) allocated memory outside the v8.

Q.34Explain the concept of stub in Node.js

Stubs are used in writing tests which are an important part of development. It replaces the whole function which is getting tested.

Q.35What is a thread pool and which library handles it in Node.js

The Thread pool is handled by the libuv library. libuv is a multi-platform C library that provides support for asynchronous I/O-based operations such as file systems, networking, and concurrency.

Q.36How to make node modules available externally

module.export

Q.37What is the default scope of Node.js application

Local

Q.38Which module is used to serve static files in Node.js

node-static

Q.39What is a Document in MongoDB

A Document in MongoDB is an ordered set of keys with associated values. It is represented by a map, hash, or dictionary.

Q.40What is the Mongo Shell

It is a JavaScript shell that allows interaction with a MongoDB instance from the command line. With that one can perform administrative functions, inspecting an instance, or exploring MongoDB.

Q.41How do you Delete a Document in MongoDB

The CRUD API in MongoDB provides deleteOne and deleteMany for this purpose. Both of these methods take a filter document as their first parameter. The filter specifies a set of criteria to match against in removing documents.

Q.42Explain the process of Sharding.

Sharding is the process of splitting data up across machines. We also use the term "partitioning" sometimes to describe this concept. We can store more data and handle more load without requiring larger or more powerful machines, by putting a subset of data on each machine.

Q.43What is a Replica Set in MongoDB

To keep identical copies of your data on multiple servers, we use replication. It is recommended for all production deployments. Use replication to keep your application running and your data safe, even if something happens to one or more of your servers. Such replication can be created by a replica set with MongoDB. A replica set is a group of servers with one primary, the server taking writes, and multiple secondaries, servers that keep copies of the primary's data. If the primary crashes, the secondaries can elect a new primary from amongst themselves.

Q.44What is Scaffolding in Express.js

Scaffolding is creating the skeleton structure of application. There are 2 way to do this: Express application generator and Yeoman

Q.45What is routing and how routing works in Express.js

Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on). Each route can have one or more handler functions, which are executed when the route is matched.

Q.46What is Middleware in Express.js

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

Q.47What Function Arguments Are Available To Express.js Route Handlers

The arguments available to an Express.js route handler function are: req (the request object), res (the response object) and next (optional, a function to pass control to one of the subsequent route handlers)

Q.48How Can I Authenticate Users in Express

Authentication is another opinionated area that Express does not venture into. You may use any authentication scheme you wish.

Q.49Which Template Engines Does Express Support

Express supports any template engine that conforms with the (path, locals, callback) signature.

Q.50How Do I Render Plain Html in Express

There's no need to "render" HTML with the res.render() function. If you have a specific file, use the res.sendFile() function. If you are serving many assets from a directory, use the express.static() middleware function.

1- Mention the purpose of MongoDB

MongoDB is defined as the document acquainted database executive which is designed for storing the data. MongoDB stores the data under the format of binary JSON, which implements the theory of anthology and documentation. MongoDB has a NoSQL database equipped with high performance, high scalability, and seamless query and indexing flexibility.

2- Mention the purpose of ExpressJS

ExpressJS can be explained as the web application framework designed to support and host NodeJS projects. ExpressJS is the open-source framework that is available under MIT. It manages the workflow between the front end and the database, facilitating the smooth and secure transfer of the data. In addition, ExpressJS is filled with excellent error-handling web design functionality for optimizing web development procedures.

3- What is the purpose of AngularJS?

It is an open-source and front-end web application managed by Google. AngularJS allows web developers to use HTML for the template language and extend the syntax of HTML to represent the components of web applications clearly and precisely.

4- List the function of NodeJS

NodeJS is an open-source cross-platform, which has a single-threaded JavaScript framework for developing server-side and networking applications.

NodeJS is called the backbone of the MERN stack. C and C++ are the programming languages used by NodeJS apart from JavaScript. It is equipped with a web server that facilitates the smooth deployment of MongoDB and cloud applications.

5- State the IDE's which are used in the development of NodeJS

For the NodeJS development, the IDE's which are used are discussed below

- Atom
- Cloud9
- Eclipse
- Komodo IDE
- JetBrainWebStorm
- JetBrains IntelliJ IDEA

6- Explain Mongoose

A mongoose is the object document mapper used for defining objects through a strongly typed schema that can be mapped to the MongoDB document. Thus, Mongoose offers a schema-based solution for modeling the application data. In addition, Mongoose had built-in typecasting, validation, query building, business logic hooks, and other features.

7- What is data modeling?

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.

8- Define REPL under NodeJS

REPL is Read Eval Print Loop, the program for accepting commands, evaluating them, and printing results. Thus, REPL does the same that Unix or Linux used to create an environment for entering commands and systems that will respond to the output.

9- Explain scope in JavaScript

Every JavaScript function represents the scope: collecting variables and rules that define the unique name access to the variable. A scope function can be operated through the code in the function. The variables are contained under a particular scope that has unique names.

10- List the difference between linear search and binary search

Linear search	**Binary search**
Under linear search, the items will be considered one at a time without performing a jumping function. Linear search classified as O(n) within the time limit for searching of the list.	On the other hand, the binary search starts searching under the middle of the item list. A binary search has been performed to identify the values greater or lesser than the desired value. Binary search is classified as O(logn).

11- Point the difference between NodeJS, AJAX, and jQuery

NodeJS, AJAX, and jQuery are the implementation of JavaScript. But are slightly different from one another. The implement NodeJS is the server-side platform that is used for the development of the client-server application. On the other hand, AJAX is the client scripting technique used to render the content without refreshing the page. jQuery is the JavaScript that complements AJAX, DOM, traversal, and looping. This is loaded with a function for promoting JavaScript development.

12- Explain Dependency Injection

This software allows injecting the service in ways that are independent of client consumption. With this, the client can be prevented from modifying dependencies underlying service changes. Dependency injection is operated for separating the client creation dependencies from the behavior, which lets you design a loosely coupled program. This function also allows the modification or tweaking of the behavior of the application through the components.

13- What is containerization?

This is the alternative to traditional hypervisor machine virtualization, which involves encapsulating the application in the container within the operating environment. Under containerization, sharing of different containers is done instead of cloning the operating system for virtual machines.

14- What is the Test pyramid, and how do you actualize the test pyramid when examining HTTP APIs?

The full-stack web applications are large and complex. They are being designed for expanding the functionalities for serving numerous requirements of the users. If the size of the full-stack code base and several users grows, the cost will also escalate. The test pyramid approach implies the radical way of thinking to estimate the numerous tests that should be created to balance the portfolio.

For actualizing the test pyramid, follow the steps given

- Incorporation of the low-level unit tests for the model.
- Inclusion of joining test, which is used for determining the working of the model.
- Conclusion of acknowledgment tests which are used for testing HTTP endpoints.

15- What is the purpose of indexing in MongoDB?

Indexes are being used for supporting and facilitating the execution of queries in MongoDB. MongoDB needs to scan every document if the index is not there and then select the appropriate form that matches the query statement. Or, if the query has an index assigned to it, MongoDB can use the index to limit the number of documents.

16- Share the difference between classes and interface in TypeScript

They both are the structures that are responsible for promoting object-oriented programming and also checking type in TypeScript. But under the class, a blueprint is there, which allows the creation of objects collection which shares the same configuration. At the same time, the interface is the collection of properties and methods which describe the thing. It doesn't provide implementation or initialization of the objects.

17- Explain decorators in typescript

A decorator is a declaration attached to the class declaration, method, accessor, property, or parameter. They are functions that take their target as the argument. Decorators will allow you to run the arbitrary code in the target execution or replace the target with a new one.

18- Define callback in NodeJS

The callback is asynchronous, which is equivalent to the function. NodeJS depends on the callbacks which are called at the culmination of the task given. After the exception of the file I/O is done, the call back function will

come to the action for passing the content file. With callback, it is ensured that there will be no blocking or waiting of the file I/O.

19- Explain cross-site scripting

The XSS or cross-site scripting is the client-side code within the malicious scripts executed in the web browser through including malicious codes in the most legitimate web page or application. Cross-site scripting can also occur when people click on the untrusted link, passing cookies and information to the hackers or attackers.

20- What is AOT?

The angular application has HTML templates along with the components. The browser cannot understand the components and HTML template directly; hence angular application needs to be compiled before running to the browsers. AOT can convert angular HTML and TypeScript code to JavaScript code during the building phase before browsers download and run the code.

21- Define grid system in CSS

The CSS system is used for stacking the content vertically and horizontally in a consistent and manageable manner. It has two components: rows and columns. The most used grid systems are Simple, Pure, Flexbox, Bootstrap, and Foundation.

22- Name the various functions performed by REPL

REPL performs the following functions

- Reading
- Evaluating
- Printing
- Looping

23- Mention the benefits of AOT

Various benefits of AOT (ahead of time) are

- With AOT, the browser can download a pre-compiled version of the application. The browser will download the executable code for rending the application immediately.
- AOT compilers inline the HTML templates and CSS style sheets within the application JavaScript externally. It eliminates separate AJAX, which requests for source files.
- AOT compiler helps detect the reports through template binding errors.

24- Define MERN stack

The term MERN stack can be defined as the collection of advancements that are based on JavaScript. MERN stack is used for creating web applications and pages.

25- State the abbreviation of MERN

The abbreviated form of MERN is

MongoDB

ExpressJS

ReactJS

NodeJS

26- Describe the features of NodeJS

Different components of NodeJS are

- Fast in code execution
- Single thread ascendible
- Usage of Library JavaScript
- No buffering

27- Explain event emitter under NodeJS

The module named event emitter in NodeJS allows creating and damaging the custom events. Under this event, the module comprises the game class emitter, which can handle and raise custom events. Also, when the emitter comes across any error, the event will have some errors. On the other hand, the event emitter will send the demonstration with the new user's name when any new user gets added. And when the user gets removed, the demonstration is named as a remote ad user.

28- State the meaning of NPM in NodeJS

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.

29- Describe non-obstructing feature under NodeJS

The feature of non-obstruction or non-blocking implies that IO is non-blocking. This hub utilizes libuv for dealing with IO in a stage nationalist manner. Under windows, non obstructing operates finish ports for Unix, which uses poll or queue and more. Also, it makes the non-obstructing demand and on-demand. Finally, it lines the inside of the occasion circle, which calls the JavaScript callback in the fundamentals of JavaScript string.

30- State the feature which is utilized under the NodeJS for importing outside libraries

The summon require is being utilized for bringing in the external libraries. Example "var HTTP=require ("HTTP")". It will stack up the HTTP library from which the single sent out for protesting through HTTP available.

31- Define occasion circle in NodeJS

For processing and handling is an outside occasion, and for changing over them for callback summons, the occasion circle is optimized. With the lines I/O calls, NodeJS can start with demand onto next.

32- Give the advantages of utilizing NodeJS

Different benefits of NodeJS are

- Using NodeJS will construct the simple method of adaptable system programs.
- Quick generally
- Simultaneously great
- Everything is asynchronous
- Never pieces

33- State the kind of API work in NodeJS

Mainly the two variants of API work under NodeJS

- Synchronous, which has to block capacities.
- Asynchronous, which have non-blocking capacities.

34- State the challenges with NodeJS

The challenge of NodeJS is that it accentuates the technical side with the touch of the test under NodeJS for one process with the string for scaling up on the multi-center server.

35- Describe non-hindering in NodeJS

Non-hindering is just like non-obstruction. It implies non-blocking in IO. The hub present there will utilize with IO in a stage septic way. Under windows, the fulfillment of ports of Unix will use epoll or kqueue and more.

36- Define the term I/O

It is the shorthand for information and yield, which will get to anything outside of the application. I/O will be stacked to the machine memory for running the program after the application begins.

37- Define occasion driven programming

Of PC programming, occasion-driven writing computer programs is the programming worldview for which the program stream is being controlled to the occasions like messages from other projects and strings. Occasion

driven programming is the application engineering system partitioned into two cases event selection and event handling.

38- Where to utilize NodeJS?

NodeJS can be utilized under the following

- Web applications
- Network applications
- Distributed applications
- General reason application

39- State the contrast of AngularJS and NodeJS

The web application improvement structure is AngularJS, and the runtime framework is NodeJS.

40- Define control stream work

It is the non-specific code that keeps running in the middle of few non-concurrent work calls.

41- Describe the advantages of React JS

The benefits of React JS are given under

- It increases the application performance through virtual DOM.
- The JSX of react JS can be easily read and write.
- Works effectively well on both client and server-side.
- React JS can be integrated with different frameworks.
- It made it easier to write UI test cases and integration with tools like JEST.

42- List the features of MongoDB

Given the features of MongoDB are

- The data model is flexible in the documents.
- It has highly scalable and agile databases.
- It is faster than traditional databases.
- It expresses query language.

43- Discuss the significance of Replication in MongoDB

The process of replication is defined as the synchronization of data across various servers. It provides redundancy which increases data availability. With the multiple data present on the different database servers, replication can protect the database from the single server loss. This process also all to recover from hardware failure and service interruptions.

44- List the limitations of the react function

The limits are stated under

- ReactJS is just a library view and not a full-blown framework.
- For those candidates who are new to web development, React offers them the learning curve.

● Integrating ReactJS with the traditional MVC framework requires additional configurations.
● The complexity of code increases with inline templating and JSX.
● Many smaller components will lead to an over-engineering boilerplate.

45- Define asynchronous API

Every API of Node is asynchronous, which means that it is non-blocking. Asynchronous API means that the server of NodeJS will never wait for API to return data. Instead, the server will move to the following API after calling it from a notification mechanism of events of NodeJS. Thus, it helps in getting the response from previous API calls.

46- When to embed one document with another in MongoDB

Consider these before embedding the documents
● Containing the relationship between entitles.
● One too many connections.
● Reasons for the performance.

47- Mention the advantages of BSON in MongoDB

Some advantage is discussed under
● BSON has been designed to become efficient in space, but it's not more effective than JSON in some cases. BSON uses more space because of the designer goals and traversability.
● BSON can easily encode and decode but uses more space than JSON for small integers.
● BSON adds up the additional data types unavailable in JSON.

48- State the solution of avoiding callback hell under NodeJS

The methods for preventing callback hell are

● Making the programming modular.

● By using the async/await mechanism.
● Through the use of the promise mechanism.
● Through the use of generators.

49- Define pure components

Pure components are the same as components ex forceps handling shouldComponentUpdate. When the props or the states are changed, the pure element will make a shallow comparison on both props and states. But component doesn't use current props and state to the next box.

50- State the meaning of reacting hooks

Hooks are the newly updated feature of React 16.8. With this, you can use state and different React features without the class. With hooks, extracting the stateful logic from the component is more accessible and can be tested independently and reused. Moreover, it allows using of stateful logic without

changing component hierarchy. With this, one can easily share the hooks amongst many components or with the community.

How MongoDB is functioning in the MERN stack?

MongoDB is a **document-oriented database** executive that is used to store data. MongoDB holds data in binary JSON format, which embodies the anthology and documentation theory. MongoDB is a NoSQL database with great performance, scalability, and flexibility in querying and indexing.

2. What is the purpose of Express JS in this technology?

ExpressJS is a web application framework that was created to facilitate and host NodeJS projects. ExpressJS is an open-source framework licensed under the **MIT license**. It coordinates the workflow between the front end and the database, ensuring that data is transferred quickly and securely. Furthermore, ExpressJS has strong error-handling web design functionality for streamlining online development processes.

3. Explain the work of ReactJS?

React produces a virtual document object model (DOM). When a component's state changes, it first performs a "**diffing**" method to determine what has changed in the virtual DOM. The second stage is reconciliation, which involves updating the DOM with the differences found in the diff.

4. How is NodeJS functioning in this technology?

NodeJS is a cross-platform open-source framework for developing server-side and networking applications that use a single-threaded **JavaScript framework**. The MERN stack's backbone is known as NodeJS. Apart from JavaScript, NodeJS uses C and C++ programming languages. It comes with a web server that allows MongoDB and cloud apps to be deployed quickly.

5. What do you know about Mongoose?

The object document mapper, or mongoose, is used to define objects using a strongly typed schema that can be mapped to a MongoDB document. Mongoose, as a result, provides a **schema-based solution** for modeling application data. Typecasting, validation, query construction, business logic hooks, and other functionality were also integrated into Mongoose.

6. How does REPL work under NodeJS?

The **Read Eval Print Loop** (REPL) is a program that accepts commands, evaluates them, and prints the results. As a result, REPL works in the same way that Unix or Linux did in terms of creating an environment for entering commands and systems that respond to the output.

7. How will you utilize Data Modeling in the MERN stack?

In the context of Mongoose and MongoDB, this word is employed. Data modeling is the process of developing a data model for the data at hand before storing it in a database. The data model depicts the **data object**, its relationship, and the rules that govern how the data is connected.

Data visualization can be portrayed while business standards, compliance, and policy are enforced through data modeling. Data modeling is used to ensure that conventions, values, semantics, security, and data quality are all consistent.

8. What distinguishes React from AngularJS(1.x)?

AngularJS (1.x) works by extending **HTML syntax** and injecting numerous components (such as Directives, Controllers, and Services) during runtime. As a result, AngularJS has strong opinions about your application's overall architecture — these abstractions are certainly valuable in some situations, but they come at the expense of flexibility.

React, on the other hand, is solely concerned with the construction of components and has few (if any) thoughts on the architecture of an application. This gives a developer a lot of flexibility in selecting the architecture they think is "optimal."

9. What do you know about Higher-Order Components?

A function that takes a component and returns a new component is known as a higher-order component (HOC). It's essentially a pattern developed from React's compositional nature. They're called "pure" components since they can accept any dynamically provided child component but don't change or copy the behavior of their input components.

const EnhancedComponent = higherOrderComponent(WrappedComponent);

HOC can be used in a variety of scenarios, as seen below.

- Bootstrap abstraction, logic, and code reuse
- Manipulation and abstraction of states
- Manipulation of objects

10. Tell us about Replication in MongoDB?

The practice of synchronizing data across several servers is known as replication. Data availability is increased and redundancy is provided via replication. Replication protects a database from the loss of a single server by storing several copies of data on various database servers. You may also recover from hardware failures and service interruptions using replication.

11. Explain Asynchronous API?

The Node.js library's APIs are all asynchronous, or non-blocking. It means that a Node.js server never has to wait for an API to return data. After accessing the next API,

the server moves on to the next one, and the notification mechanism of Node.js Events assists the server in receiving the response from the previous API call.

12. Tell me about Reconciliation?

When a component's **props** or **state** change, React compares the newly returned element to the previously rendered one to determine whether an actual DOM update is required. React will update the DOM if they are not equal. This is referred to as reconciliation.

13. What do you know about CallBack Hell?

As a return parameter, the asynchronous function expects callbacks. A callback hell condition occurs when numerous asynchronous functions are chained together.

14. When should we use MongoDB to embed one document inside another?

You should consider embedding documents for the following reasons:

- Includes inter-entity relationships
- One-to-many connections are the most common type of relationship.
- Reasons for poor performance

15. Tell me the advantages of BSON over JSON in MongoDB?

BSON is supposed to be space-efficient, however, it isn't always more efficient than JSON. BSON needs considerably more space than JSON in some circumstances. Another of BSON's design aims is traversability, which explains why. BSON adds "additional" information to documents, such as string and subobject lengths. This speeds up traversal.

BSON is also made to encode and decode quickly. Integers, for example, are kept as **32 (or 64) bit integers** and do not require parsing to and from the text. For small integers, this takes up more space than JSON, but it is significantly faster to parse.

16. Explain how Node JS handles child threads?

In essence, Node.js is a single-threaded application. The developer is not given access to child threads or thread management mechanisms. Although Node.js does launch child threads for certain activities like asynchronous I/O, these run in the background and do not execute any JavaScript code or disrupt the main event loop.

There are tools available to enable threading functionality in a Node.js application, such as the ChildProcess module.

17. How Pure Components Work?

PureComponent is identical to Component, with the exception that it takes care of the **shouldComponentUpdate** method for you. PureComponent does a shallow comparison on both props and state when they change. Out of the box, Components will not compare current props and state to the next. As a result,

anytime **shouldComponentUpdate** is triggered, the component will re-render by default.

18. Tell us about React Hooks?

In **React 16.8**, hooks are a brand-new feature. They make it possible to use state and other React capabilities without having to create a class. Hooks allow you to abstract stateful logic from a component so that it may be tested and reused separately. Hooks allow you to reuse **stateful functionality** without having to change the hierarchy of your components. This makes it simple to distribute Hooks across several components or to the community.

19. What is Key and why should you use it in your lists?

When generating lists of elements, a key is a particular string attribute that must be included. React uses keys to figure out which items have changed, been added to, or been removed.

We frequently use IDs from your data as keys, for example.

const todoItems = todos.map((todo) =>

<li key={todo.id}>

{todo.text}

**
**

);

You can use the item index as a key as a last resort if you don't have reliable IDs for rendered items:

const todoItems = todos.map((todo, index) =>

<li key={index}>

{todo.text}

**
**

);

20. What do you mean by a Blocking Code?

If an application needs to wait for an I/O operation before continuing its execution, the code responsible for waiting is referred to as blocking code.

21. How does Node.js handle concurrency if it is single-threaded?

Node gives programmers access to a single thread, allowing them to write code quickly and efficiently. For different I/O activities such as File, DNS, Network calls, and so on, Node uses many POSIX threads internally.

When Node receives an I/O request, it creates or utilizes a thread to complete the operation, and then it pushes the result to the **event queue**. On each such event,

the **event loop** runs and checks the queue, adding the queue result to the execution stack of Node if it is empty.

22. Explain Shadow DOM?

The main goal of Shadow DOM is to encapsulate the implementation. A single custom element can mix more or less complex logic with a more or less sophisticated DOM.

An import and **<my-app>/my-app>** tag can be used to bring a whole web application of any complexity to a page. Simpler reusable and composable components, such as **<date-picker>/date-picker>**, can likewise be implemented as custom elements with their internal representation concealed in the shadow DOM.

23. What are the smart component and dumb component?

Smart components manage their state or are connected to the Redux store in a Redux context.

Dumb components are fully controlled by the props they receive from their parent and have no state of their own.

24. What is the usage of createRef?

createRef is always going to make a **new ref**. When building a class-based component, you'd usually place the ref in an instance property (e.g. this.input = createRef()). This option isn't available in a function component.

25. Explain the procedure of the Event loop in Node JS?

Although JavaScript is single-threaded, the event loop allows Node.js to conduct non-blocking I/O operations by transferring operations to the system kernel whenever available. Every I/O requires a callback, which is then pushed into the event loop to be executed. Because most current kernels are multi-threaded, they can handle many background operations. When one of these tasks is completed, the kernel notifies Node.js, allowing the appropriate callback to be added to the poll queue and executed later.

26. Tell us about Prop drilling and the ways to avoid it?

When developing a React application, it's common to need a deeply nested component to access data from a component far higher in the hierarchy. The most straightforward method is to simply pass a prop from one component to the next in the hierarchy, from the source component to the deeply nested component. This is referred to as prop drilling.

Prop drilling's main downside is that it makes components that should not be aware of data overly complicated and difficult to maintain.

A typical technique to avoid prop drilling is to leverage React context. This enables the definition of a data-supply Provider component, as well as the consumption of context data by nested components via either a Consumer component or a **useContext hook**.

27. How will you query with the “like” operator in MongoDB?

The below query can be simply converted as a MongoDB query as

select *

from users

where name like ‘%m%’

db.users.find({“name”: /.*m.*/})

// or

db.users.find({“name”: /m/})

28. Do you know about Occasion Driven Programming?

The programming viewpoint of occasion-driven creating computer programs is one in which the program stream is controlled by events such as notifications from other projects and strings. The application engineering framework of event-driven programming is divided into two parts: **event selection and event handling.**

29. Modify this code into Async/Await.

function asyncTask() {

return functionA()

.then((valueA) => functionB(valueA))

.then((valueB) => functionC(valueB))

.then((valueC) => functionD(valueC))

.catch((err) => logger.error(err))

}

Answer: async function asyncTask() {

try {

const valueA = await functionA()

const valueB = await functionB(valueA)

const valueC = await functionC(valueB)

return await functionD(valueC)

} catch (err) {

logger.error(err)

}

}

30. What is an event emitter in NodeJS?

In NodeJS, the event emitter module allows you to create and destroy custom events. The game class emitter, which may handle and raise custom events, is included in this event. In addition, if the emitter encounters an issue, the event will contain some errors. When a new user is added, the event emitter, on the other hand, will send the demonstration with the new user's name. When the user is no longer present, the demonstration is referred to as a remote ad user.

1. **What is your favorite programming language, and why it is your favorite?**

The full-stack developer usually know

- HTML & CSS
- Python
- JavaScript
- MEAN

So as a Full Stack Developer, you should be able to know these programming languages but also be able to choose your favorite language that you understand in-depth, and you are comfortable with it. So that you can easily demonstrate when asked.

2. So what was the latest thing you learned?

This question asked by the interviewer is to understand what all you know, and they will indeed explore the credibility of your CV. Also, they would like to see what you did in the recent six months. You can explain about the assignment or project and let them know what bugs you faced and what solution you found and resolved the bugs. Also, you can explain the websites you visited what you learned, and discuss the recent trends. This will help you out and let the interviewer know what you did in the current six months span.

3. What technologies and programming languages do you require from start to finish for developing a project?

It is the question of how well the interviewer can analyze your willingness to start the work. This is a question of discussion. A competent and proficient complete stack developer is a convenient way to distinguish it from one who is a rookie. Therefore, you must be careful to respond.

4. What is pair programming? Have you ever done it?

Pair programming is the two programmers who will be working on the same terminal. The two programmers need to involve in the programming need to consider the essential elements of extreme programming. Considering your experience in pair programming, you must answer accordingly at your convenience.

5. What is CORS?

CORS is a shortened form for cross-origin resource sharing and enables the sharing of resources from a multitude of sources, as the name implies. CORS is the procedure used in different domains to access various web resources. Due to CORS, web scripts could be more freely connected with external content of the original domain, and thus, better integration among web services can also be achieved.

When accessing resources from domain2.com to domain1.com, then domain1.com needs a simple cross-origin request, and it will be done. These resources can be anything that is an image, CSS file, or any other. This all has some significant security restrictions, and also the built-in behavior of the browser has limited access to the cross-origin HTTP request.

6. Are you aware of design patterns?

You should be aware of the bugs and errors that are commonly faced while designing any web application. If you gain expertise is very clear, then you must use the force the employer to gain trust by explaining the experience which you have a done coding.

7. What is multithreading?

This question tests the computer architectural knowledge of the candidate who is applying for the full stack developer job profile. The multithreading process is very beneficial as it improves the performance of the CPU. This is supported by the operating system and done by executing multiple threads or strategies. Multithreading can manage a program at a time by more than one user or handle several requests by the same user. It is achieved primarily by running several processes that the operating system can sustain.

8. What is Continuous Integration?

Continuous integration helps in automating device development, testing, and deployment. If a single person or a whole team develops software projects, generally use continuous integration as a hub to ensure crucial steps such as unit testing are automatic, instead of manual tasks. This is why the developers can quickly deploy codes during development time. This is mainly used several times a day to integrate principles. The essential advantage is the identification and eventual detection of errors.

9. What are you coding currently?

Whether for your company or your leisure purpose, a person who loves technology will always work with it.

You will stand out here if you’re one of the programmers who code for skills development.

10. Explain inversion of control?

In contrast with conventional control flow, IoC (Inversion of Control) inverts the flow of control. In IoC, a generic control flux is given to custom-written portions of a computer program. In conventional programming, the custom coding that communicates the program's intent, calls on reusable libraries to carry out a generic function. Still, in reverse control, it is the frame to call in the custom or task-specific code. The software architecture has the same design and reverses the traditional procedural programming method.

11. What is long polling?

Long polling is an effective method for creating a very stable server connexion without the use of WebSocket or Server Side Events protocols. Node.js use this technique as the Next Development Model.

In other words, Long Polling operates at the top of the conventional client-server model. Here, the customer sends the request, and the server responds until the connexion is open as it contains new and unique information.

As soon as the server answers, a request can be submitted to the client, and when data is available, the server will return a query. It functions when the client application stops and the server ends requests.

12. How do you keep up-to-date on new industry trends?

One right way to address this question is to explain your involvement and understanding you got through ongoing learning in the discussion you attended with friends, colleagues, or online. Also, if you have personal projects on where you used your talents, then it is an excellent time to demonstrate that too. You can even speak about the webinars or forums you attend regularly.

13. Can you tell us an example as to when and how you have handled an inefficiency in the code of somebody else?

The interviewer will often know how relaxed and up to mark you point out defects or glitches in other programs to find out how proficient a candidate is with programming.

14. Give an example of a project and the technology you have been working on. How have you picked these?

This helps to understand the strategy of a candidate and also offers an understanding of their efficiency in identifying the perfect toolset. It would help if you defined as well as dig into the specifics when thinking about the purpose of using a specific toolset. Show your ability to play in the creation of a web app with the frontend and backend. It's all right to prove that you have more expertise on the one hand than on the other but to show you've got the potential to do both things.

15. Describe some examples of a web application that you have built, and how did you do it?

The interviewer will get an understanding of how you think and how you deal with methods. The response you are giving should be straightforward and detailed why you would have developed it at the front or back end.

16. How can you build your CSS and JavaScript so that other developers can work with them more efficiently?

Developers at the front end must develop codes created by skilled employees or collaborate as part of a team. Complete code is divided into section, and each code and section have detail comments so anyone can understand what the code about is

17. Explain the key difference between GraphQL and REST?

The key difference between GraphQL and REST APIs is that a REST API is a network-based software architecture idea. GraphQL, by comparison, is a query, specification, and collection of tools that work with HTTP over one single endpoint.

18. In all of your designs, what is the most significant error you did? How have you corrected it?

It's just not practical to function on and continue working on the technology. The response to this question should be frank, speak about a mistake and how serious it was, and then talk about your knowledge from the error and explain the way you wanted it to mitigate the damages done.

19. What's the most bizarre programming challenge you have come across recently?

You will talk about the new bug you encountered and explain how you handled it and how you overcame it. Inform the interviewer of the information obtained during the bug solution and how it could first be prevented.

20. What standards would you consider for SEO?

SEO means Search Engine Optimization. SEO will demand the creation of an optimized position that fits well for search engine rankings. The criteria to be used for SEO include the use of alt tags containing images, the company's social media must be linked to the web, using the XML sitemap, eliminating broken links, etc.

21. How can someone optimize their website?

The following points should be recalled for the optimization of the website:

- Analyze all the data on your web
- Conduct detailed research on keywords
- Making long content rich in value
- SEO on-page optimization
- SEO off-page optimization

- Optimize the mobile website
- Accelerate the page
- Get your quality backlinks
- Avoid CSS and JavaScript inline
- Gzippering
- Reduce the code
- Using srcset for sensitive images
- Browser caching is to be leveraged

22. Explain the term Full-stack developer?

A full-stack web developer is a person who can create both client and server applications.

- Besides, mastering HTML and CSS, he/she even knows how to:
- Browser program (such as JavaScript, jQuery, Angular, or Vue)
- Writing code for a server (like using PHP, ASP, Python, or Node)
- Code a database (such as SQL, SQLite, or MongoDB)

Full-stack developers are responsible for the frontend and backend creation of a website (or web app); they are responsible for the architecture, database, clients, and device engineering. Full-stack developers are famous for their diverse skills and comprehensive knowledge of web development.

23. Explain the responsibilities of a full stack developer?

A full-stack web developer is a person who can create both client and server applications.

Besides mastering HTML and CSS, he/she even knows how to:

- Browser program (such as JavaScript, jQuery, Angular, or Vue)
- Writing code for a server (like using PHP, ASP, Python, or Node)
- Code a database (such as SQL, SQLite, or MongoDB)
- Responsibility of Full Stack Developers:
- We are developing the architecture of the front-end website.
- We are designing interactions between users on web pages.
- We are developing back-end site software.
- I am creating functionality servers and databases.
- Ensure cross-platform optimization for cell phones.

- Ensure the responsiveness of applications.
- Working on web design features with graphic designers.
- We see through the project from the design to the final product.
- Design and development of APIs.
- I was responding to both technological and customer needs.
- I am staying up to date with advances in web applications and programming languages.

Full-stack developers are responsible for the frontend and backend creation of a website (or web app); they are responsible for the architecture, database, clients, and device engineering. Full-stack developers are famous for their diverse skills and comprehensive knowledge of web development.

24. Explain the term front end?

Anything with which the user interfaces is the front end of a program or website. From the user's point of view, the front end is synonymous with the user interface. … Websites need to work well on different platforms and screen sizes, which is why modern web development usually requires responsive design.

25. What development languages are used for server-side coding?

The server languages, rather than the client languages, are the programming languages used to conduct work on the server before submission to the browsers. For that website, just limited server-side information is provided.

The Web Platform Docs, as mentioned in the FAQ item, will not concentrate on servers. However, since they are so relevant, we provide a fundamental introduction and links for each language to other, more definite places.

- Perl
- PHP
- Python
- Ruby
- Java
- JavaScript

26. What DBMS technologies used for full-stack development?

Some of the DBMS technologies used for full-stack development are as follows

- MySQL
- Oracle

- SQL Server
- MongoDB

27. Explain the software stack?

A software stack is a set of independent components that enable the implementation of an application together. The elements, including an operating system, architectural layers, protocols, runtime environments, databases, and calling functions, are stacked in a hierarchical position one over the other. The lower levels of the hierarchy typically deal with hardware, and the higher levels of the order carry out complex tasks for the terminal user. Several complicated instructions passing through the stacks allow components to interact directly with the application.

28. Explain the LAMP stack?

LAMP-a a well-known web development software stack. LAMP (Linux, Apache, MySQL, PHP) The LINUX operating system is the least level of Stack's hierarchy. The scripting language is the highest layer of the order in this case PHP. (Note: the "P" can also be used in Python or Perl programming languages). LAMP stacks are common as they all have open source components and the Stack can be used with commodities hardware. A LAMP stack is loosely connected, unlike monolithic software stacks which are typically tightly linked and designed for a specific operating system. This means they have proved to be interchangeable and frequently used together, although the components were not initially designed to work together. Nowadays, almost all Linux distributions have LAMP modules.

29. Explain the MERN stack?

For the four core technologies that make up the Stack, MERN stands for MongoDB, Express, React, and Node.

MongoDB-archive of documents

Node.js- Web platform for Express(.js)

JavaScript's system on the client side of React(.js)

Node(.js)-the first web server with JavaScript

MERN is one of a variety of variants of the MEAN (MongoDB Express English Node) stack, where React.js is substituted for the standard Angular.js frontend system. Other versions like MEVN (MongoDB, Express, Vue, and Node) and JavaScript can operate in any frontend.

The middle (application) tier consists of express and node. Express.js is a Web platform on the server-side and Node.js a common and efficient server platform for JavaScript. No matter what variant you pick, ME(RVA)N provides the perfect way to work with JavaScript and JSON.

30. Explain MEAN stack?

MEAN (MongoDB, Express, Angular, and Node) — a collection of tools to boost software creation, famous for helping to remove frequent language barriers. The base of a MEAN stack is MongoDB, a NoSQL data storage text. The Express and Angular HTTP server is the basis for the JavaScript frontend. Node, the medium for server-side scripting, is the highest layer of the Stack.

31. Explain the application server?

Software frameworks for the construction of application servers are application server frameworks. Both the web application and server environment can be created in an application server framework.

A comprehensive service layer model is included in an application server framework. It contains a range of components that the software developer can access by means of a standard platform API. These components generally operate in the same environment as their web servers for web applications, and their main task is to facilitate the creation of dynamic pages. The various application servers do more than build web pages. To enable developers to focus on the implementation of business logic, they implement the services like clustering, failover, and load balancing.

32. Explain referential transparency?

The aspect of functional programming is referential transparency. It has been used to substitute the expression in a program but does not change the final result.

33. What are some design patterns?

Some design pattern is as follows

- UI
- GUI
- UX
- Prototype and database principles.

34. Explain the debugging process of a complex program?

The given problem must be analyzed before debugging. The inputs and outputs should always be checked. Data sets for information and work shall be checked as test data for possible inputs and outputs. Correction means that the same error must be avoided during the next phase and project.

35. Can you tell me what the newest trends for full-stack development are?

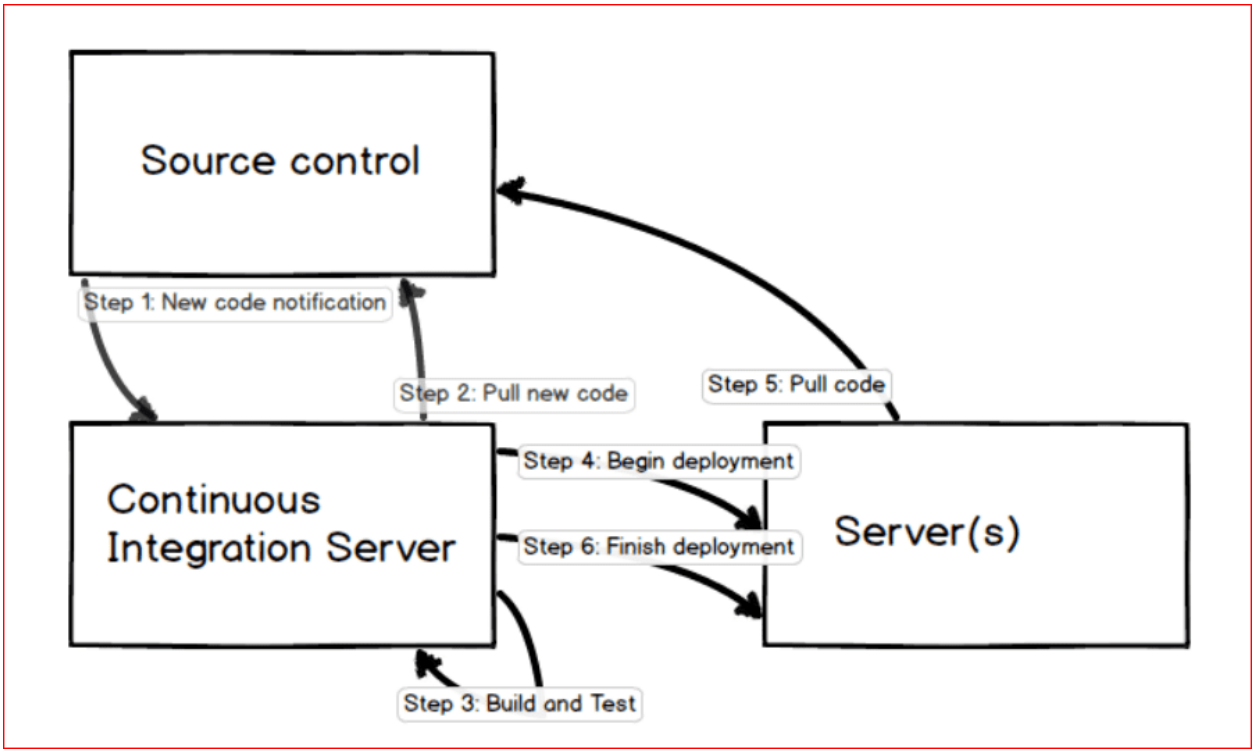
Several of these trends entail fully compliant extensions, improvements to JavaScript programming, JavaScript functionality, etc.

36. What are the essential skills needed to become a full-stack developer apart from technical skills?

Full Stack Developer has a lot of capabilities. Here are the abilities of Full Stack Developers not up for negotiation!

- CSS / HTML. Whereas HTML stands for the markup language of Hypertext, CSS means Cascading Style Sheets.
- JavaScript.
- Git and GitHub.
- Backend languages.
- Web architecture.
- HTTP and REST.
- Database storage.
- Basic design skills.

37. What are the steps in Continuous Integration?



The server continues to integrate and evaluate the code. If all checks pass, the deployment process starts with the continuous integration server. The new code is pulled down to the server where it is deployed. Finally, reboot services and associated deployment activities complete the deployment process.

A continuous integration server can be organized and deployments structured in many other ways. This was only an example of a straightforward configuration.

38. How will you define Continuous Integration? Continuous integration automates device development, testing, and deployment. Software projects, whether they are generated by one single person or by whole teams, usually use continuous integration as a hub to ensure that crucial measures such as unit testing are automated and not manual.

39. Explain the 3-tier/3 layer model?

For any application, three levels of the model consist of three layers. A presentation layer that relates to the front end of the user interface, business layer, and the backend portion is used to validate the data. The third layer is the layer of the database, which deals with data storage.

40. What is semantic HTML?

The webpage loop is described using HTML. Semantic HTML focuses on the importance of semantics of the info posted on the website. Important text can be found on the website and ranked by search engines.

41. Explain the CSS Box model?

The model CSS Box is used to evaluate the web page content layout. Each feature that is displayed on the webpage is shown as a rectangular box. The contents to be viewed on-screen are margin, side, lining, contents borders, specific sizes, and colors.

42. What is bootstrap in full-stack development?

Bootstrap is an open-source kit with HTML, JS, and CSS content developer tools. It can prototype an idea and build the app with the help of SaaS variables, mixtures, grids, prefabricated modules, and plugins.

43. Why is REST important in the HTTP protocol?

REST is incredibly simple and builds on frameworks that already exist. To achieve targets, it uses existing HTTP features. It does away with the need for new standards, technology, frameworks, etc.

44. How is REST different from SOAP?

There are two APIs that vary from the following:

REST is a non-official architectural style. SOAP is an accepted standard protocol.

REST uses a wide variety of standards, including HTTP, JSON, URL, and XML, while SOAP primarily uses XML and HTTP.

45. What is Git?

Git helps developers to monitor improvements made to the base by means of their version control system. It is necessary to understand its essential features correctly to get the latest out of it.

46. What are the benefits of Full-stack JavaScript?

It offers a variety of advantages including code reuse, shared libraries, templates, and models, easiness to research, rapid development, no compilation, wide distribution, etc.

47. Explain the responsive web design?

CSS and HTML are both concerned. It is used to redimension the website dynamically. It looks better on any computer than on a website. Tablet, laptop. Phone, tablet.

48. Explain CSS icons?

These are available in CSS scalable and CSS customizable vector libraries. Some libraries include bootstrap icons, fantastic fonts, and google icons.

49. Name one software registry library?

The most important library in the world is the npm app registry. The packages are 800, 000. It is also used for remote administration.

50. Explain the difference between architectural and design patterns?

A reusable solution to common software architecture problems is an architectural design.

A model is a reusable approach to software design problems.

51. Explain the issues that are addressed by architectural patterns?

The issues that are addressed by architectural patterns are as follows

- High availability
- Performance
- Security
- Scalability
- Testing
- Deployment
- Maintainability
- Technology Stack

52. Name mostly used architectural designs?

The most used architectural patterns are as follows

- MVC pattern
- Master-slave pattern
- Layered pattern

- Model view presenter
- Monolithic architecture
- REST
- Event-driven architecture

53. What different types of design patterns?

The following kinds are available:

Creational pattern: Used for constructing objects, singletons, prototypes, abstract factory, singletons, and more.

Structural patterns: they promote the design by providing a straightforward way to establish connections between different entities such as adapters, façades, bridges, decorators, proxy models, etc.

Behavior patterns: These are used to classify patterns of contact between objects.

54. Is a full-stack developer different from a software engineer?

The full-stack developer has experience in the front and back end. They know customer-side programming languages, history, operating systems, databases, project management. They are all familiar with the stages. The software engineer designs write and check software so that it works correctly without errors. Software engineer, You only have one degree of understanding.

55. What is meant by application architecture?

Application Architecture is responsible for the complicated creation of software. It needs a detailed understanding of the code structure, databases, file separation, computational activities, media file hosting.

56. Explain Sass?

It is labeled as Amazing StyleSheet Syntactically. It is the preprocessor for CSS and gives language elegance. This enables the application of variables, mixtures, nesting rules, inline imports, etc. The SaaS supports the arrangement of large stylesheets. Sass helps you to run tiny stylesheets quickly.

57. Explain Mixin?

Mixin is the code block to be reused on the Web for the community of CSS declarations.

58. How is resetting CSS different from normalizing it?

Default browser element styling resetting strips. Standardization is used to retain standard types and does not include un-style objects. It is used to fix errors, as well.

59. Explain JavaScript coercion?

Converting one incorporated into another is known as coercion. It is implied or explicit in two ways. Direct coercion involves explicit conversion using the data form. Automatic conversion of data types happens in tacit coercion.

60. How is null different from undefined?

0 is a non-value object although undefined is a type.

61. Explain the use of external CSS at the place of inline?

The typically misleading impact of Inline CSS on website results. HTML weights are more by using inline scripts. The use of external CSS reduces the size of the HTML and makes it easy to render the website.

62. Explain anonymous functions?

These are anonymous functions. The names of the variable are invoked automatically.

63. What is HTML DocType?

It's a browser command written on the page bookmark version language. The HTML tag is used. DocType means DTC, i.e. description of the document type.

64. How is the call different from applying?

Apply is used as an argument array to call a method—the calluses detailed parameter listing.

65. What is visibility: hidden?

It means that the original room is invisible but still occupies it.

66.What is display: none?

It means it is secret, and it takes no space.

Explain what is DevOps ?

Answer

DevOps is a newly emerging term in IT field, which is nothing but a practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals. It focuses on delivering software product faster and lowering the failure rate of releases.

What are the success factors for Continuous Integration?

Answer

- Maintain a code repository
- Automate the build
- Make the build self-testing
- Everyone commits to the baseline every day

- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

What is Callback Hell?

Answer

The asynchronous function requires callbacks as a return parameter. When multiple asynchronous functions are chained together then callback hell situation comes up.

What is the difference between git pull and git fetch?

Answer

In the simplest terms, git pull does a git fetch followed by a git merge.

- When you use pull, Git tries to automatically do your work for you. **It is context sensitive**, so Git will merge any pulled commits into the branch you are currently working in. pull **automatically merges the commits without letting you review them first**. If you don't closely manage your branches, you may run into frequent conflicts.
- When you fetch, Git gathers any commits from the target branch that do not exist in your current branch and **stores them in your local repository**. However, **it does not merge them with your current branch**. This is particularly useful if you need to keep your repository up to date, but are working on something that might break if you update your files. To integrate the commits into your master branch, you use merge.

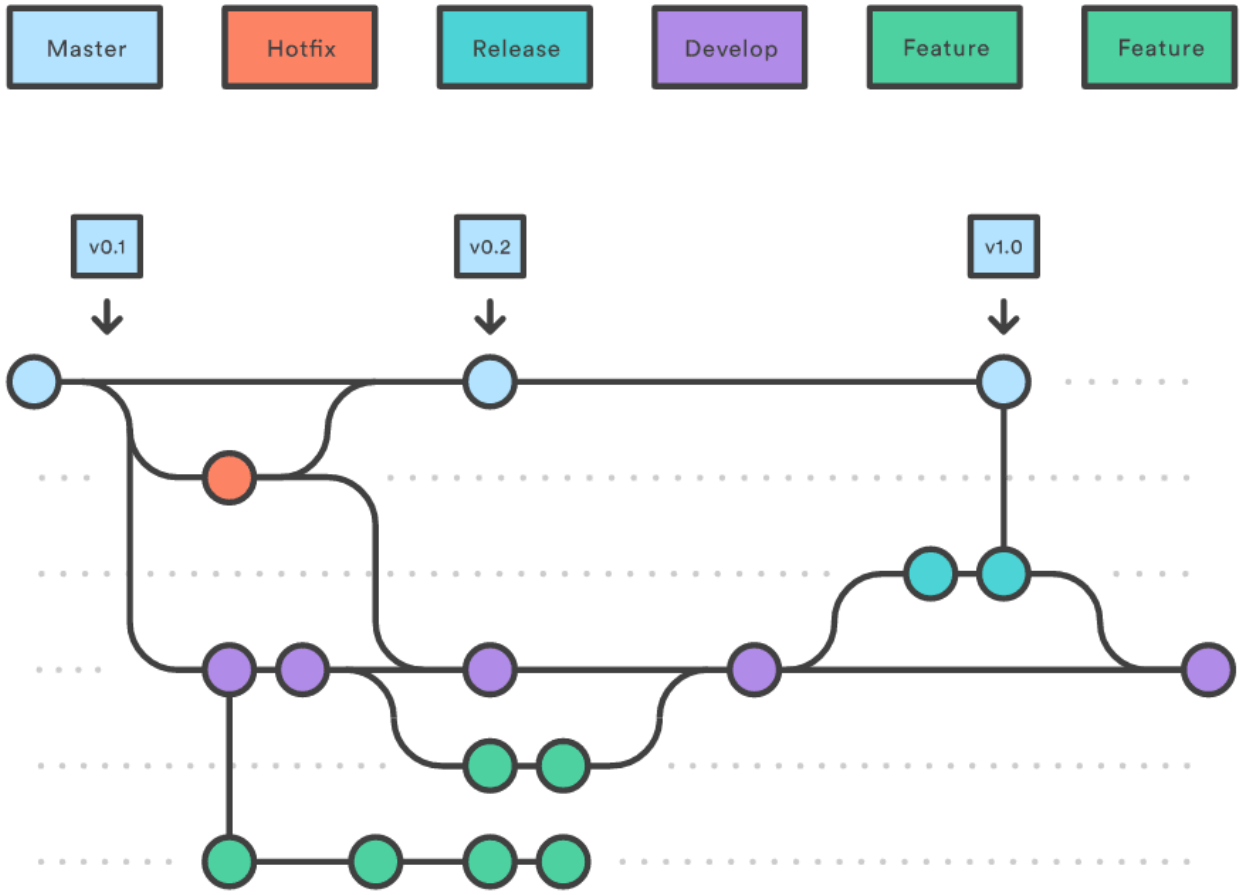
Could you explain the Gitflow workflow?

Answer

Gitflow workflow employs two parallel long-running branches to record the history of the project, master and develop:

- **Master** - is always ready to be released on LIVE, with everything fully tested and approved (production-ready).
- **Hotfix** - Maintenance or "hotfix" branches are used to quickly patch production releases. Hotfix branches are a lot like release branches and feature branches except they're based on master instead of develop.
- **Develop** - is the branch to which all feature branches are merged and where all tests are performed. Only when everything's been thoroughly checked and fixed it can be merged to the master.

- **Feature** - Each new feature should reside in its own branch, which can be pushed to the develop branch as their parent one.



Explain a use case for Docker

Answer

- Docker a low overhead way to run virtual machines on your local box or in the cloud. Although they're not strictly distinct machines, nor do they need to boot an OS, they give you many of those benefits.
- Docker can encapsulate legacy applications, allowing you to deploy them to servers that might not otherwise be easy to setup with older packages & software versions.
- Docker can be used to build test boxes, during your deploy process to facilitate continuous integration testing.
- Docker can be used to provision boxes in the cloud, and with swarm you can orchestrate clusters too.

Explain the main difference between REST and GraphQL

Answer

The main and most important difference between REST and GraphQL is that GraphQL is not dealing with dedicated resources, instead everything is regarded as a graph and therefore is connected and can be queried to app exact needs.

What are the advantages and disadvantages of using use strict?

Answer

'use strict' is a statement used to enable strict mode to entire scripts or individual functions. Strict mode is a way to opt into a restricted variant of JavaScript.

Advantages:

- Makes it impossible to accidentally create global variables.
- Makes assignments which would otherwise silently fail to throw an exception.
- Makes attempts to delete undeletable properties throw (where before the attempt would simply have no effect).
- Requires that function parameter names be unique.
- this is undefined in the global context.
- It catches some common coding bloopers, throwing exceptions.
- It disables features that are confusing or poorly thought out.

Disadvantages:

- Many missing features that some developers might be used to.
- No more access to function.caller and function.arguments.
- Concatenation of scripts written in different strict modes might cause issues.

Overall, I think the benefits outweigh the disadvantages, and I never had to rely on the features that strict mode blocks. I would recommend using strict mode.

What are the differences between ES6 class and ES5 function constructors?

Answer

Let's first look at example of each:

```
// ES5 Function Constructor
function Person(name) {
  this.name = name;
}
```

```
// ES6 Class
class Person {
  constructor(name) {
    this.name = name;
  }
}
```

For simple constructors, they look pretty similar.

The main difference in the constructor comes when using inheritance. If we want to create a Student class that subclasses Person and add a studentId field, this is what we have to do in addition to the above.

```
// ES5 Function Constructor
function Student(name, studentId) {
  // Call constructor of superclass to initialize superclass-derived members.
  Person.call(this, name);

  // Initialize subclass's own members.
  this.studentId = studentId;
}

Student.prototype = Object.create(Person.prototype);
Student.prototype.constructor = Student;

// ES6 Class
class Student extends Person {
  constructor(name, studentId) {
    super(name);
    this.studentId = studentId;
  }
}
```

It's much more verbose to use inheritance in ES5 and the ES6 version is easier to understand and remember.

What do you think of AMD vs CommonJS?

Answer

Both are ways to implement a module system, which was not natively present in JavaScript until ES2015 came along. CommonJS is synchronous while AMD (Asynchronous Module Definition) is obviously asynchronous. CommonJS is designed with server-side development in mind while AMD, with its support for asynchronous loading of modules, is more intended for browsers.

I find AMD syntax to be quite verbose and CommonJS is closer to the style you would write import statements in other languages. Most of the time, I find AMD unnecessary, because if you served all your JavaScript into one concatenated bundle file, you wouldn't benefit from the async loading properties. Also, CommonJS syntax is closer to Node style of writing modules and there is less context-switching overhead when switching between client side and server side JavaScript development.

I'm glad that with ES2015 modules, that has support for both synchronous and asynchronous loading, we can finally just stick to one approach. Although it hasn't been fully rolled out in browsers and in Node, we can always use transpilers to convert our code.

What is Event Loop in Node.js?

Answer

Generally speaking, the **event loop** is a **mechanism that waits for and dispatches events or messages in a program**. In Node.js, event loops are the central control flow constructs. For example, every time a request is about to be handled, it’s put on the event loop and processed as soon as it’s ready to be processed. Node, instead of doing it on its own, delegates the responsibility of handling the system. Because of such behaviour, Node is not actively waiting for this task to finish and can handle other requests in the meantime. The event loop makes Node.js faster and more efficient than other technologies

What will be the output of the following code?

Problem

```
var x = { foo : 1};
var output = (function() {
  delete x.foo;
  return x.foo;
})();

console.log(output);
```

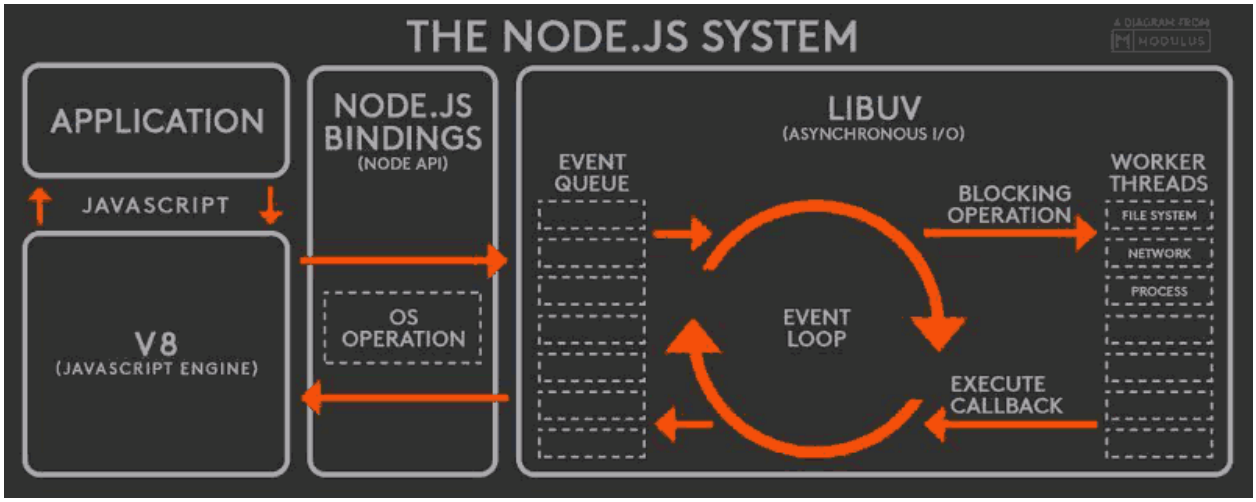
Answer

Above code will output undefined as output. delete operator is used to delete a property from an object. Here x is an object which has foo as a property and from self-invoking function we are deleting foo property of object x and after deletion we are trying to reference deleted property foo which result undefined.

What's the Event Loop?

Answer

The event loop is what allows Node.js to perform non-blocking I/O operations — despite the fact that JavaScript is single-threaded — by offloading operations to the system kernel whenever possible.



Every I/O requires a callback - once they are done they are pushed onto the event loop for execution. Since most modern kernels are multi-threaded, they can handle multiple operations executing in the background. When one of these operations completes, the kernel tells Node.js so that the appropriate callback may be added to the poll queue to eventually be executed.

What's the difference between a blue/green deployment and a rolling deployment?

Answer

- In **Blue Green Deployment**, you have TWO complete environments. One is Blue environment which is running and the Green environment to which you want to upgrade. Once you swap the environment from blue to green, the traffic is directed to your new green environment. You can delete or save your old blue environment for backup until the green environment is stable.
- In **Rolling Deployment**, you have only ONE complete environment. The code is deployed in the subset of instances of the same environment and moves to another subset after completion.