By OnlineInterviewQuestions.com

Mean Stack Interview Questions

Practice Best Mean Stack Interview Questions and Answers

MEAN (short form for MongoDB, Express.js, AngularJS, and Node.js) stack is a public and open-source JavaScript software bundle for building dynamic websites and web applications. Since all elements of MEAN stack aid programs are written in JavaScript, applications are written in any one of the languages for server-side and client-side execution environments. The components of the MEAN stack are higher-level including a web application presentation layer and not including an OS layer.

So, practice below the **Best Mean Stack Interview Questions** that are very helpful for the freshers & experienced candidates. These **Mean Stack Interview Questions and Answers** are very popular and asked various times in Mean Stack Interview. apart from this, you can also download below the **Mean Stack Interview Questions PDF**, completely free.

Q1. What is the Mean Stack?

Mean stack is combination four popular JavaScript-based technologies MongoDB, Express, AngularJS and Node that allows developers to develop complete web applications.

Q2. What is Express?

Express
is lightweight web framework for Node.js.It provides robust and scalable solutions to build single and multi-page web application. Express is inspired by Sinatra (Ruby framework).

Q3. Explain Mongoose?

Mongoose is a MongoDB Object Data Modeling (ODM) library for MongoDB and NodeJS.It provides a straight-forward, schema-based solution to model your application data. It includes built-in typecasting, validation, query building, business logic hooks and more, out of the box.

Q4. How AngularJS is different from NodeJS.

The purpose of Angular and Node Js is totally Different.
AngularJS is front-end framework that deals with UI and client side of an application while NodeJs is a runtime environment useful in building server-side applications.

Q5. How to install express?

Run below command to install express

```
npm install express --save
```

Q6. Explain Routing in Express.js?

Routing is a mechanism used by frameworks to decide how a URL/endpoint is responded/handled by the server. Express provides an excellent way to handle applications routing. Below is basic code to handle routing in Express.

```
var express = require('express')
var app = express()
// respond with "hello world" when a GET request is made to the homepage
app.get('/', function (req, res) {
  res.send('hello world')
});
```

Q7. What are the features of Node.js?

Node.js is a single-threaded but highly scalable system that utilizes JavaScript as its scripting language. It uses asynchronous, event-driven I/O instead of separate processes or threads. It is able to achieve high output via single-threaded event loop and non-blocking I/O.

Q8. Enlist the major IDEs that are commonly used for Node.JS development?

sdafasdfasdsdfadsfasdfasdfsdfadsfadsfasdfsdfdfdfdfdfdfdf

Q9. What are the difference between Classes and Interfaces in TypeScript?

In **TypeScript**, a class is a blueprint for an object, while an interface is a contract for the shape of an object.

A class can have properties and methods, and can also include a constructor for initializing the object. Classes can also have access modifiers (such as "public" or "private") to control the visibility of its members.

on the other hand, an interface only defines the structure of an object, including its properties and methods, but does not provide any implementation. It can be used to define a contract that multiple classes can implement. Interfaces do not have access modifiers.

Additionally, a class can implement multiple interfaces, but can only inherit from one other class.

Q10. What are closures and how are they useful?

In JavaScript, a closure is a function that has access to the variables and functions defined in its outer scope, even after the outer function has returned.

Here is an example of a closure in JavaScript:

```
function outerFunction() {
  let outerVariable = 'I am the outer variable';
  function innerFunction() {
    console.log(outerVariable);
  }
  return innerFunction;
}
let myClosure = outerFunction();
myClosure();  // logs "I am the outer variable"
```

Q11. How do i implement modules natively in javascript?

JavaScript does not have native support for modules, but you can use various tools and libraries to achieve the same functionality.

One popular way to implement modules in JavaScript is through the use of the 'module' and export keywords in the ECMAScript (ES)6 standard. You can use the 'export' keyword to make properties and methods available to other parts of your code, and the 'import' keyword to consume those exports in other files.

For example, in a file named 'math.js':

```
export function add(a, b) {
    return a + b;
}
export function multiply(a, b) {
    return a * b;
}
```

And in another file, 'main.js':

```
import { add, multiply } from './math';
console.log(add(1, 2)); // 3
console.log(multiply(2, 3)); // 6
```

Another popular way is by using module bundlers like webpack or browserify, which will bundle all the modules together in a single file that can be loaded by the browser.

You can also use a package manager like npm or yarn to manage your dependencies and to include modules from npm registry.

Note that some older browsers may not support ES6 modules natively, so you may need to use a transpiler like Babel to convert your code to a version of JavaScript that is supported by older browsers.

Q12. Enlist the some common design patterns in OO Javascript?

Some common design patterns in object-oriented JavaScript are as follows -

Constructor pattern: This pattern creates objects with a constructor function that sets the initial state and methods of the object.

Module pattern: This pattern creates a singleton object that provides a private and public interface to a set of functions and data.

Prototype pattern: This pattern creates objects that inherit properties and methods from a prototype object.

Revealing module pattern: This pattern creates a singleton object that provides a private and public interface to a set of functions and data, with the added benefit of only exposing the methods and properties you want to.

Factory pattern: This pattern creates objects without specifying the exact class of object that will be created.

Command pattern: This pattern encapsulates a request as an object, separating the request from the object that handles it.

Q13. What are the major differences between linear searches and binary searches.

The **major differences between linear searches and binary searches** are -

Linear search:

Linear search checks each element in the data set in sequence, starting from the first element, until it finds the element it is looking for. It is also called sequential search. It has a time complexity of O(n) where n is the number of elements in the data set. It is simple to implement and can be used with any type of data set, but it is not efficient for large data sets.

Binary search:

Binary search is used on a sorted data set and it works by dividing the data set in half and checking whether the element it is looking for is in the first half or the second half. It then continues to divide the half that contains the element in question and checks again, until it finds the element. It has a time complexity of O(log n) which is much faster than linear search for large data sets. The downside is that the data set needs to be sorted and it is only applicable for numerical or string data that can be ordered.

In summary, Linear search is simpler to implement, but less efficient for large data sets. While binary search is more efficient for large data sets, but requires the data set to be sorted and is only applicable for ordered data.

Q14. What is a callback? Expain the pros & cons of callback in node js.

Callback is a function that is passed as an argument to another function and is executed after some event or action occurs. The function that accepts the callback is often referred to as the "parent" function, and the function that is passed in as an argument is referred to as the "callback" function.

In Node.js, callbacks are used extensively to handle asynchronous operations. For example, when reading a file or making an HTTP request, the Node.js runtime will not wait for the operation to complete before moving on to the next line of code. Instead, it will pass a callback function that will be executed when the operation is completed.

Pros of Callback in Node.js:

- They allow you to handle asynchronous operations, which makes your code non-blocking and efficient.
- They provide a clear separation of concerns, allowing you to handle the success and failure cases of an operation separately.
- They are widely used in Node.js and are supported by many built-in modules, allowing you to take advantage of pre-existing functionality.
- They are compatible with most versions of Node.js, and therefore they can run on most environments.

Cons of Callback in Node.js:

- They can lead to callback hell, which is a situation where you have nested callbacks that are hard to read and maintain.
- They don't provide a way to cancel a callback or a way to check if a callback has been called.
- They don't support the concept of return values, and it becomes difficult to handle errors when there are multiple asynchronous operations.

Q15. What is the purpose of containerization?

Containerization is a method of packaging and deploying applications in a way that makes them easy to run and scale. It involves packaging an application and its dependencies into a container, which is a standardized unit of software that includes everything the application needs to run, including the code, runtime, system tools, libraries, and settings.

Containers are self-contained and lightweight, so they can be easily moved between different environments, such as development, staging, and production. They are also isolated from the host system and from other containers, so they do not interfere with each other or with the host system.