**What is Node Js?**

Node JS is Single Threaded Language   
Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

Node. js uses callbacks, being **an asynchronous platform**, it does not wait around like database query, file I/O to complete

Node js is a JavaScript Runtime built on Chrome V8 JavaScript Engine. Node js is a JavaScript running on the Server. Node js Used Asynchronous Programming

JavaScript is client Side Scripting Language but with the help of node js we run JavaScript on the server.

Browser doesn’t understand JavaScript for every browser have JavaScript engine which is used to convert JavaScript into machine understandable language. In Google chrome we have V8 Engine, Mozilla have spider Monkey, and safaris have Nitro

**What is nodejs Worker Thread?**

Worker Threads in NodeJs is **useful for performing heavy JavaScript tasks**. With the help of threads, Worker makes it easy to run JavaScript codes in parallel making it much faster and efficient. We can do heavy tasks without even disturbing the main thread

History of Node js:-

In 2009 an idea comes in the mind of Ryan Dahl (G.E) that why not runs JavaScript outside the browser. So he took v8 engine and embedded in c++ program and called it Node.exe later on become nodejs.

**What is Express:-**

* Express is Web Framework for Node js which used Create Web Server
* The Basic Functionality of Web Server is to Accept Request and Send the Response
* Express helps to organize your application into a MVC architecture
* M:- model say that to have all the model related data like database,
* C: there is controller in which all the business logic should be preset here
* V: there is all that things present which is visible to the user
* Most important thing that express does is graceful error handling it handle the error very gracefully that’s why the server will not crash.
* Express is best candidate of creating rest apis

**JavaScript History:-**

JavaScript was developed by Brendan Eich, who was a Netscape programmer. Brendan Eich developed this new scripting language in just ten days in the year September 1995. At the time of its launch, JavaScript was initially called Mocha. After that, it was called Live Script and later known as JavaScript.

**What the difference is between == and ===?**

The == operator checks equality only whereas === checks equality, and data type, i.e., a value must be of the same type

**What are/is callbacks? With example?**

* Node js usually used Asynchronous programming language. It means that they don’t wait around the thing like file I/O to finish.

e.g. without using callbacks

=> As we all know that node js usually used asynchronous programming it means that the doesn’t wait to execute that code who takes more time to complete, they execute another code who takes less time than that in this situation there will be chances of to crash the code that take more time. To handle this situation we used callbacks.

* let data;

function fetchData () {

setTimeout (() => {

data = {proId: 1001, proName: ‘Orange'}

}, 2000)

}

function displayData () {

console.log (data);

console.log ('end here')

}

console.log ('start here');

FetchData ();

displayData ();

Output of this program is:-

start here

undefined

end here

Because of asynchronous nature of node js and to prevent this situation callbacks are in the seen. See the following example.

e.g. using Callback

let data;

function fetchData (cb) {

setTimeout (() => {

        data = {proId: 1001, proName: ‘Orange'}

        cb ();

    }, 2000)

}

console.log ('start here');

fetchData (function displayData (){

    console.log (data);

 console.log ('end here')

});

Output of this program is:-

start here

{proId: 1001, proName: ‘Orange’}

end here

**What is callback hell?**

* This is a big issue caused by coding with complex nested callbacks. Here, each and every callback takes an argument that is a result of the previous callbacks. In this manner, the code structure looks like a pyramid, making **it difficult to read and maintain**.
* The use of callbacks Hell **makes the code difficult to write and maintain**. It also increases the difficulty of identifying the flow of the application, which is an obstacle when it comes to making debug, hence the famous name to this problem: Callback Hell

**What are promises with example and why it comes?**

A Promise in NodeJS is similar to a promise in real life. It is an assurance that something will be done. Promise is used to keep track of whether the asynchronous event has been executed or not and determines what happens after the event has occurred. It is an object having 3 states namely:

* **Pending:**Initial State, before the event has happened.
* **Resolved:** After the operation completed successfully.
* **Rejected:** If the operation had error during execution, the promise fails.

Advantage:-   
Promises are **used to avoid callback hell in writing Node.** **js**, which will eventually happen if you want to call multiple asynchronous calls in

Sequence.

See the following example which using promise

E.g. Using Promises

function fetchData () {

return new Promise ((resolve, reject) => {

setTimeout(() => {

   let data = {proId: 1001, proName:'Orange'}

    resolve(data)

}, 2000)

})

}

console.log ('start here');

fetchData ().then ((data) => {

    console.log (data);

}).catch ((error) => {

    console.log (error);

}).finally (() => {

    console.log ('Programs end here');

})

**Async await with example and why it comes?**

* [**Promises**](https://www.educative.io/edpresso/what-are-promises-in-javascript) were added in JavaScript to handle asynchronous operations easily. The **async/await** is syntactical sugar, added in ES7, that facilitates working with promises. Async/await helps with writing completely synchronous-looking code while performing asynchronous tasks
* await is used to wait for a promise to resolve or reject, and can only be used inside an asynchronous function.

E.g.

app.post ('/student', async (req, res) => {

    try {

        const user = new Student (req.body);

        const superUser = await user.save()

        res.status(201).send(superUser)

    } catch (error) {

        res.status(400).send(error)

    }

})

**What is next() and return next()?**

* next()**:**It will run or execute the code after all the middleware function is finished**.**
* return next()**:**By using return next it will jump out the callback immediately and the code below return next() will be unreachable

**What is middleware?**

* Middleware functions are **functions that have access to the request object (req)**, the response object (res), and the next middleware function in the application's request-response cycle.
* It is the mediator which is responsible for request and response
* Basically there are two types of Middleware
* Predefined middleware
* Costume middleware
* Predefined middleware are app.use, app.listen
* Costume middleware are that created by ourselves
* Like auth middleware when we applied session management
* Multer that is fileupload middleware

**What is Body parser?**

* Body-parser is the Node. js body parsing middleware. It is **responsible for parsing the incoming request bodies in a middleware before you handle it**. It used for form handling, it is used for multipart form handling

**What cors?**

* CORS stands for **Cross-Origin Resource Sharing**. It allows us to relax the security applied to an API. This is done by bypassing the Access-Control-Allow-Origin headers, which specify which origins can access the API.
* It helps us to use api anywhere

**What is logger?**

Logger is used to see how many people are wisiting our website or app and also we how many ips are going to see our site. Popular tools are used see this kind of information are DATADOG,SOLARWINDS LOGGLY

**What is Hoisting?**

* Hoisting is Phenomenon in JavaScript by which we can access the variable even before they/you have initialized it. Or u put some value in it. You can access these variable without getting any error

**What is lexical scope or lexical environment?**

* Lexical environment is the local memory along with lexical environment of its parent
* Lexical is term means in hierarchy or in sequence
* E.g

Function a( ){

Var b = 10 ;

C();

Function c(){

}

}

a ();

console.log (b)

* In the above example the c function is lexically inside a function

**What is Temporal Dead Zone?**

* Temporal dead Zone is the time, since when let variable was hoisted and till it is initialize in it the between that is known as temporal dead zone . from hoisting to value initialize in it that phase In between that is known as temporal dead Zone
* All the callback function which comes from promises is comes in the micro task queue.

**What is a closure?**

* A Function Along with its lexical Scope is bind together or bundled together is known as closures.

**What is the difference between let var and const?**

* We can access the var variable even before they initialized
* But in case of let and const we cannot access them before they initialized because they are stored in temporal zone
* If we access them before they initialized and it gives the reference error that the variable cannot access before it initialized
* In case of const it is mandatory to declared as well as initialized at same time
* In case of let and const they are stored in separate memory space not in global space like var
* Const is more strict than let

**What is the Features of ES6 in JavaScript?**

* Let and Const keyword
* Arrow Function
* Promises
* Default Parameters
* Template laterals

**What is routing in Express.js?**

routing is the Mechanism used by framework to decide how URL/endpoint is respond/handled by the server

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).

**What is Aggregation in nodejs?**

. It collects values from various documents and groups them together and then performs different types of operations on that grouped data like sum, average, minimum, maximum, etc to return a computed result

**Why node js is single threaded?**

         Node js is based on single threaded architecture due to the reason that is runs GOOGLE ‘S V8 ENGINE which is a single threaded architecture, it’s built to run exactly one thread per JavaScript execution context

         Over the time, single threaded architecture of Node provides much better performance one thread per request architecture

**What is API ?**

**API** stands for application programming interface which allows two applications to communicate with each other over the internet through various devices.

**what are webservices?**

Web services provides a common platform that allows multiple applications built on various programming languages to have the ability to communicate with each other

**What is Rest Apis?**

Rest api stands for Representational State Transfer. They are stateless Backend. Stateless backend means no matter with state. Rest ape stored or return the data in the form of JSON.

**What is the difference between PUT and PATCH?**

When we want to update the single data we used PATCH. But I want to update whole body or Object then I Used PUT method.

**Types of relation in mongo dB?**

**Why we use express. Static?**

The **express.static()** function is a built-in middleware function in Express. It serves static files and is based on serve-static

**Parameters**: The root parameter describes the root directory from which to serve static assets.

**Return Value**: It returns an Object.

**Why we write usrlencoded in express?**

The **express.urlencoded ()** function is a built-in middleware function in Express. It parses incoming requests with urlencoded payloads and is based on body-parser.

**Parameter:** The options parameter contains various property like extended, inflate, limit, verify etc.  
 **Return Value:** It returns an Object.

**What is error first callback?**

**Error-First Callback**in Node.js is a function which either returns an error object or any successful data returned by the function.

1. The first argument in the function is reserved for the error object. If any error has occurred during the execution of the function, it will be returned by the first argument.
2. The second argument of the callback function is reserved for any successful data returned by the function. If no error occurred then the error object will be set to null.

**What is http Module?**

Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

Use the createServer () method to create an HTTP server:

**What is mean by module.exports?**

Exports, we can export functions, objects, and their references from one file and can use them in other files by importing them by require() method. Purpose: The main purpose of module. **Exports is to achieve modular programming**

**Different Types of Modules in Nodejs?**

* 1. Core Modules
  2. Local Modules
  3. Third Party Modules

**What is lodash.js?**

**What is Dockers?**

**What is microservices?**

**What is reddis cache?**

**Event emmiters in nodejs?**

**Streams in nodejs?**