<<<<<<<<<<<<<<<<<<<< node js >>>>>>>>>>>>>>>>>>>>

1.What is Node.js?

=>Node. js is an open-source, cross-platform JavaScript runtime environment and library for running web applications outside the client's browser.

2. How does Node.js differ from other server-side technologies like Java or PHP?

=>Node. js is different from existing server-side frameworks because it is based on asynchronous events via JavaScript callback functionality and uses the JavaScript as a programming language. Moreover, everything inside Node. js runs in single thread.

3.What is NPM, and why is it used in Node.js?

=>npm stands for Node Package Manager. It's a library and registry for JavaScript software packages. npm also has command-line tools to help you install the different packages and manage their dependencies

4.Explain the concept of event-driven programming in Node.js?

=> events => signal send button click er upar particular code execute (node js a eta api through button thake na)

eventsemitter => events ke generate korake bole events emitter ,its basically class

ex =>

```
const express = require("express");

const EventEmitter = require("events");

const app = express();

const event = new EventEmitter();

let count = 1;

event.on("home", () => {

  console.log(" home event called", count++);

});
```

```
event.on("about", () => {
  console.log("  about event called", count++);
});

app.get("/", (req, res) => {
  res.send("api called");
  event.emit("home");
});

app.get("/about", (req, res) => {
  res.send("api about called");
  event.emit("about");
});

const port = 9444;
app.listen(port, () => {
  console.log(`server is running at: @http://127.0.0.1:${port}`);
});
```

5.. What is the role of the package.json file in Node.js projects?

=>For Node. js projects, the package. json file provides a simplified way to manage a project's metadata and dependencies.

The file ensures a project always has current information about the libraries and tools it needs to work correctly.

6.How can you handle errors in Node.js?

=> Error object.

Try… catch.

Throw.

Call stack.

Effective function naming.

Asynchronous paradigms like promise.

ex=>

7. What is the purpose of the module.exports and exports objects in Node.js?

=>module.exports is an object in a Node.js file that holds the exported values and functions from that module.

8.Describe the role of the core modules in Node.js.

=> 1. basic features prothom thekei thake , 1. data base connect 2. files create ,3. api call

example console.log o core module, http,

9.How can you create a simple HTTP server using Node.js?

=>

Node js server Creation ?>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

=>

```
const http = require('http')
const server = http.createServer((req,res)=>{
   res.end('here is my server ')
})
const port = 9874
server.listen(port,()=>{
console.log(`my server is running at :@ http://127.0.0.1:${port}`);
})
```

10.Explain the difference between synchronous and asynchronous code in Node.js.

=> // // asyncronous

```
// const fs = require('fs')

//  fs.readFile('index.txt','utf-8',(error,data)=>{
//     if (error) {
//         console.log("here is no file");

//     }else{
//         console.log(data);
//     }
// })

// console.log("hello");

// syncronous ////////////////////////////////////

const fs = require('fs')
const data = fs.readFileSync('index.txt','utf-8')
console.log(data);
console.log('hello');
```

11..What is the purpose of callback functions in Node.js? Give an example.

```
=>setTimeout(() => {
   console.log('Callback as Arrow Function');
}, 10
```

12.What is the purpose of the require function in Node.js?

=> In NodeJS, require() is a built-in function to include external modules that exist in separate files. require() statement basically reads a JavaScript file, executes it, and then proceeds to return the export object.

13.How can you read and write files in Node.js?

=>

```
 const fs = require('fs')
fs.writeFile('new.txt','here is new file ',(err)=>{
   console.log("file created");
   console.log(err);



})

fs.readFile('new.txt9','utf-8',(error,data)=>{
   if (error) {
      console.log("here is no file Created");

   }
   else{
      console.log(data);
   }
})
```

14.Explain the concept of streams in Node.js and provide an example use case.

=> The Streams API allows JavaScript to programmatically access streams of data received over the network and process them as desired by the developer

15.What is middleware in the context of Express.js?

=>

A request handler with access to the application's request-response cycle is known as middleware.

It's a function that holds the request object, the response object, and the middleware function.

Middleware can also send the response to the server before the request.

The next middleware function is commonly represented as a variable named next.

Simply middleware is a function that can only be applied using routes.

We can access and modify request and response data using middleware.

16.How can you handle form data in an Express.js application?

=>To get started with forms, we will first install the body-parser(for parsing JSON and url-encoded data) and multer(for parsing multipart/form data) middleware. var express = require('express'); var bodyParser = require('body-parser'); var multer = require('multer'); var upload = multer(); var app = express(); app.

17.What is the purpose of the body-parser middleware in Express.js?

=>

Basically what the body-parser is which allows express to read the body and then parse that into a Json object that we can understand. When receiving a POST or PUT request, the request body might be important to your application.

18.What is the purpose of the next function in Express.js middleware?

=>

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.

19.How can you secure a Node.js application?

=>

Make use of HTTPS. HTTPS is the HTTP protocol's secure version. ...

Make use of Authentication. Authentication is a method of verifying a user's identity. ...

Restrict access. ...

Validation of input. ...

Implement security best practices. ...

User Models. ...

Post Models. ...

Register the endpoint.

20. What are some popular frameworks and libraries used with Node.js?

=> express, vue

21.Default port of Mongodb

=> 27017

22.Why we use underscore / lodash in our application?

=>

Underscore. js is a utility library that is widely used to deal with arrays, collections and objects in JavaScript. It can be used in both frontend and backend based JavaScript applications. Usages of this library include filtering from array, mapping objects, extending objects, operating with functions and more

23.How can you create a simple express server using Node.js? >>>>>>>>>>>>>

=>

```
const express = require('express')
const app = express()

app.get("/", (req,res)=>{
   res.send("hello")
})

const port = 9444
app.listen(port,()=>{
   console.log(`server is running at: @http://127.0.0.1:${port}`);
})
```

24. How can you create a simple mongo db connection in Node.js? >>>>>>>>>>>>

=>

```
const mongoose = require('mongoose');

const connection_string =



let option = {
   auth: {
      username: process.env.DB_USERNAME,
      password: process.env.DB_PASSWORD,

   },
   useNewUrlParser: true,
   useUnifiedTopology: true
}

module.exports = () => {
   try {
      mongoose.connect(connection_string, option);
      console.log("database Connected Successfully");
   }catch(err) {
      console.log(err);
   }
}
```

25.Difference between find() and findOne()>>>>>>>>>>>>>>>>>>>>

=> The findOne() returns first document if query matches otherwise returns null.

The find() method does not return null, it returns a cursor.

=> .find() a all data chole ase & curser return ex : .limit(2)

.find({age:11}) => jader jader 11 ache tara asbe

.findone({age:11})=> 1ta asbe data=. return null not curser return

=> read all elements find no arguments under

=> findOne a particular 1 data

26. Can you please demonstrate the flow of login and registration using JWT >>>>>>>>>>>>

=>

27. How can we upload a single file in node js?>>>>>>>>>>>>>>>>

```
=>  const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        console.log("myFile", file);
        cb(null, "./public/uploads")
    },
    filename: (req, file, cb) => {
        const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() * 1E9)
        cb(null, file.fieldname + '-' + 'Anupam_Mishra' + uniqueSuffix + path.extname(file.originalname))
    }
});

// const maxSize = 1024;

const uploads = multer({
    storage,
    fileFilter: (req, file, cb) => {
        if (file.mimetype === 'image/jpg' || file.mimetype === 'image/jpeg' || file.mimetype === 'image/png') {
            cb(null, true);
```

```
        }
        // else if(file.mimetype === 'application/msword' || file.mimetype === 'application/pdf'){
        //     cb(null, true);}

         else {
            cb(null, false);
            return cb(new Error('only jpg, jpeg, png are allowed'))
        }
    },
    // limits: maxSize
})
```

28.How can we upload multiple files in node js?>>>>>>>>>>>>>>>>>>>>

=>

30. request a data form browser? >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

=>

```
const express = require('express')
const app = express()

app.get("/", (req,res)=>{
    console.log("data send by Browser ", req.query.name);
   res.send(` "Hello" ${req.query.name}`)

})
port = 9444
app.listen(port,()=>{
```

```
  console.log(`server is running at: @http://127.0.0.1:${port}`);
})
```

31. why nodejs is single threaded ?

=> node js works on the single threaded model to support the asyncronous processing which provides

high performance and efficiency to its applications under high amount of load

32. what is rest api ?

=> A RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources.

33. what is bycript ?

=> The bcrypt is a password hashing technique used to build password security. It is used to protect the password from hacking attacks because of the password is stored in bcrypted format

34. what is jwt?

=> A JSON Web Token, popularly known as JWT, is an open standard that defines a compact way for securely sharing information between two parties: a client and a server.

35.what is dotenv ?

=> A JSON Web Token, popularly known as JWT, is an open standard that defines a compact way for securely sharing information between two parties: a client and a server.

36. what is package.json?

=> it stores the details of our projects related to coding like-project name, version of project,

git repository, commands list of installed packages

37. what is models in mongoose and schemae?

=> schema => a mongoose schema defines the structure of the document , default values, validators etc

=> A mongoose model is a wrapper on the mongoose Schema ,a mongoose model provides an interface to the database for creating,querying,updating

deleting the records etc

<<<<<<<<<<<<<<<<< javascript 2 >>>>>>>>>>>>>>>>>>>>>>>>

1. Callback Function =>

a callback function is a function passed into another function as an arguments

ex:

```
function func1(){
   console.log("hiw i am anupam ");
}

function func2(callback){
   console.log("my age is 22");
   callback()
}

func2(func1)
```

2. all Types of functions ====>

1. function statement ==> // hosting support // also known as Function decleration

```
 function a (){
   console.log("hello");
}
a()
```

2. function expression // not support hoisting

=>

```
let x = function(){
   console.log("anupam");
}

x()
```

3.function Declaration

4.anonomous function

=> function without a name doesnot have identity return syntax error

uses=> we can use as an value

5. named functtion expression // create inside not global // create local variable

```
=> let x = function xyz(){
   console.log(xyz);
}
x()
```

6. difference between parameter and arguments?

=>

7. first class function ?

=> same as callback

8. Higher order function ?

=> Higher function are functions that take other functions as arguments or return functions as their results.

ex=>

9. call, apply & bind ?

=> Call invokes the function and allows you to pass in arguments one by one.

Apply invokes the function and allows you to pass in arguments as an array. Bind returns a new function,

allowing you to pass in a this array and any number of arguments.

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Array methods >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

1. filter => Creates a new array with every element in an array that pass a test

2. find => Returns the value of the first element in an array that pass a test

3. forEach => Calls a function for each array element

4. map => Creates a new array with the result of calling a function for each array element

5. pop => Removes the last element of an array, and returns that element

6.push => Adds new elements to the end of an array, and returns the new length

7.Reduce => Reduce the values of an array to a single value (going left-to-right)

8.reverse => Reverses the order of the elements in an array

9. shift => Removes the first element of an array, and returns that element

10. slice => Selects a part of an array, and returns the new array

11. sort => Sorts the elements of an array

12.splice => Adds/Removes elements from an array

13.unshift => Adds new elements to the beginning of an array, and returns the new length

14 join => Joins all elements of an array into a string

10. closure example=>

```
function hello(name) {

  var message = "hello " + name;

  return function hello() {

    console.log(message);
```

```
  };

}

//generate closure

var helloWorld = hello("World");

//use closure

helloWorld();
```

11. explain import and exports =?

=>

12. this keyword in javascript ?

=> The Keyword ‘this’ in JavaScript is used to call the current object as a constructor to assign values to object properties.

13. What is the difference between Call and Apply? (explain in detail with examples)

Call

Call uses arguments separately.

Example:

```
function sayHello()

{

  return "Hello " + this.name;
```

```
}

var obj = {name: "Sandy"};

sayHello.call(obj);

// Returns "Hello Sandy"
```

Apply

Apply uses an argument as an array.

Example:

```
function saySomething(message)

{

  return this.name + " is " + message;

}

var person4 = {name:  "John"};

saySomething.apply(person4, ["awesome"]);
```

14. Explain Hoisting in javascript. (with examples)

Hoisting in javascript is the default process behavior of moving declaration of all the variables and functions on top of the scope where scope can be either local or global.

Example 1:

```
hoistedFunction();  // " Hi There! " is an output that is declared as function even after it is called

function hoistedFunction(){

  console.log(" Hi There! ");

}
```

Example 2:

```
hoistedVariable = 5;

console.log(hoistedVariable); // outputs 5 though the variable is declared after it is initialized

var hoistedVariable;
```

15. currying in JavaScript (with examples)

In JavaScript, when a function of an argument is transformed into functions of one or more arguments is called Currying.

Example:

```
function add (a) {

  return function(b){

    return a + b;

  }
```

```
}


add(3)(4)
```