OPERATION

INTERVIEW QUESTIONS WITH ANSWER

Python

1. What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It's high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development

Javascript

1. What is Javascript?

JavaScript (**JS**) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB, and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. Read more about JavaScript.

- **What are the features of Javascript?**
    - JavaScript is a lightweight, interpreted programming language.
    - JavaScript is designed for creating network-centric applications.
    - JavaScript is complementary to and integrated with Java.
    - JavaScript is complementary to and integrated with HTML.
    - JavaScript is open and cross-platform.

2. What is Node.Js?

Node.js is an open-source server environment;
Node.js is a cross-platform JavaScript runtime environment that allows developers to build server-side and network applications with JavaScript.

3. What is Express?

Express is a minimal and flexible **Node**.**js** web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of **Node** based Web applications..

4. Require:

require() is not part of the standard JavaScript API. But in Node.js, it's a built-in function with a special purpose: to load modules. Modules are a way to split an application into separate files instead of having all of your applications in one file.

- **Built-in function:**
  built-in function. A function that is built into an application and can be accessed by end-users. For example, most spreadsheet applications support a built-in SUM function that adds up all cells in a row or column.

5. App:

Each **app**.**use**(middleware) is called every time a request is sent to the server. **app**.**use**() **used** to Mounts the middleware function or mount to a specified path, the middleware function is executed when the base path matches. In the above code **app**.**use**() mount the path on '/ads' to adsRouter.**js**

6. Body-parser:

body-parser is a middleware that **extracts** the entire body portion of an incoming request and exposes. **body-parser** module parses the JSON, buffer, string, and URL encoded data submitted using an HTTP POST request.

7. JWT(Jsonewebtoken):

The claims in a **JWT** are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.

Authentication and Authorization:-

authentication is the process of verifying who a user is, while **authorization** is the process of verifying what they have access to. Comparing these processes to a **real-world example**, when you go through security in an airport, you show your ID to authenticate your identity.

What is the difference between JWT and OAuth?:

So the real difference is that **JWT** is just a token format, **OAuth 2.0** is a protocol (that may use a **JWT** as a token format). Firstly, we have to differentiate the **JWT** and **OAuth**. Basically, **JWT** is a token format. ... **OAuth** uses server-side and client-side storage.

8. **Knex:**

Knex.js is a JavaScript query builder for relational databases including PostgreSQL, MySQL, SQLite3, and Oracle. It can be used with callbacks and promises. It supports transactions and connection pooling.

9. **Routes:**

Routing defines the way in which the client requests are handled by the application endpoints.

10. **What is the CRUD method**

Within computer programming, the acronym CRUD stands for create, read, update and delete. These are the four basic functions of persistent storage. Also, each letter in the acronym can refer to all functions executed in relational database applications and mapped to a standard HTTP method, SQL statement or DDS operation.

11. **Server.listen()**

The **Node**.**js** framework can be **used** to develop web **servers** using the 'Http' module. The **application** can be made to **listen** on a particular port and send a response to the client whenever a request is made to the **application**. The 'request' module can be **used** to get information from websites

12. **What is Git?**

Git is a free, open-source distributed version control system tool designed to handle everything from small to very large projects with speed and efficiency. It was created by Linus Torvalds in 2005 to develop Linux Kernel. Git has the functionality, performance, security and flexibility that most teams and individual developers need.

13. **Git vs Github?**

Git is a version control system, a tool to manage your source code history. **GitHub** is a hosting service for **Git** repositories. So they are not the same thing: **Git** is the tool, **GitHub** is the service for projects that use **Git**.

14. **Module.export**

The **module**.**exports** or **exports** is a special object which is included in every **JS** file in the Node.**js** application by default. The **module** is a variable that represents the current **module** and **exports** is an object that will be exposed as a **module**.

15. **What is the Framework?**

A web application framework is a combination of libraries, helpers, and tools that provide a way to effortlessly build and run web applications.

16. <u>What is EJS?</u>

EJS or Embedded Javascript Templating is a templating engine used by **Node**.**js**. Template engine helps to create an HTML template with minimal code. Also, it can inject data into HTML template at the client-side and produce the final HTML.

17. <u>Static and Dynamic:</u>

In general, **dynamic** means energetic, capable of action and/or change, or forceful, while **static** means stationary or fixed. In computer terminology, **dynamic** usually means capable of action and/or change, while **static** means fixed.

18. <u>API:</u>

API stands for Application Program Interface, which can be defined as a set of methods of communication between various software components. In other words, an **API** allows the software to communicate with other software.

19. <u>Restful API:</u>

A **RESTful API** is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. ... REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST leverages less bandwidth, making it more suitable for internet usage.

20. <u>Npm:</u>

npm is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command-line client, also called npm, and an online database of public and paid-for private packages called the npm registry.

21. <u>What are endpoints?</u>

An **endpoint** is one end of a communication channel. When an **API** interacts with another system, the touchpoints of this communication are considered **endpoints**. For **APIs**, an **endpoint** can include a URL of a server or service. ... The place that **APIs** send requests and where the resource lives are called an **endpoint**.

22. <u>What is controlar?</u>

AngularJS application mainly relies on **controllers** to control the flow of data in the application. A **controller** is defined using the ng-**controller** directive. A **controller** is a **JavaScript** object that contains attributes/properties, and functions.

23. <u>Event loop:</u>

This means that everything that happens in **Node** is the reaction to an **event**. A transaction passing through **Node** traverses a cascade of callbacks. Abstracted away from the developer, this is all handled by a library called libuv which provides a mechanism called an **event loop.**

24. Callback:

A **callback** is a function that is to be executed after another function has finished executing — hence the name '**call back**'. More complexly put: In **JavaScript**, functions are objects. ... Any function that is passed as an argument is called a **callback** function
Exa:-

```
Function show(a){
      console.log("Hello" + a);
}
Function hide(callback){
      Var a = "NavGurukul";
      callback(a);
}
hide(show);
```

25. Callback Hell:

Callback Hell, also known as Pyramid of Doom, is an anti-pattern seen in code of programmers who are not wise in the ways of asynchronous programming. ... It consists of multiple nested callbacks which makes code hard to read and debug.

26. Promises:

Most of the issues with nested callback functions can be mitigated with the use of **promises** and generators in **node**.**js**. A **Promise** is a value returned by an asynchronous function to indicate the completion of the processing carried out by the asynchronous function.

A **promise** is an object which can be returned synchronously from an asynchronous **function**. It will be in one of 3 possible states: Fulfilled: onFulfilled() will be called (e.g., resolve() was called)

- pending - The initial state of a promise.
- fulfilled - The state of a promise representing a successful operation.
- rejected - The state of a promise representing a failed operation.

Exa:-

```
var RoomClean = new Promiss((resolve, reject) =>{
      let isClean = true;
      if(isClean){
            resolve("Room is clean");
      }else{
            reject("Room is not clean")
      }
});
RoomClean.then((fromResolve) =>{
      console.log(fromResolve);
}).catch((fromReject) =>{
      console.log(fromReject);
})
```

27. ECMAscript (es7 / es6):

ECMAScript (or **ES**) is a scripting language specification standardized by Ecma International in **ECMA-262** and ISO/IEC 16262. It was created to standardize JavaScript, so as to foster multiple independent implementations. JavaScript has remained the best-known implementation of ECMAScript since the standard was first published, with other well-known implementations including JScript and ActionScript.[2] ECMAScript is commonly used for client-side scripting on the World Wide Web, and it is increasingly being used for writing server applications and services using Node.js.

28. Synchronous/Asynchronous:

★ Synchronous:

Synchronous basically means that you can only execute one thing at time.
When we execute a program synchronously then the program executes line by line means step by step or from up to down.
Python is a **synchronous** language.

★ Asynchronous:

Asynchronous means that you can execute multiple programs at a time and you don't have to finish executing the current thing in order to more one to the next one.
When we execute the program asynchronously then the program does not execute line by line, so we can move to step if step one will not complete then it will left step one and work on step two.
Javascript is an **asynchronous** language

29. Blocking/Non-blocking:

★ Blocking:

Blocking means a snippet (part) of code that executes line by line does not matter how much time it takes.
Suppose that if we have 3 lines of code and first-line takes 5 sec to execute, the second line takes 3 sec to execute and the third line takes 4 sec to execute, then according to blocking system first of all first line of code will execute then second, then third and then fourth.
In the **blocking** system, we wait for each line output until we get first lines output we can't move to the next line.

★ Non-blocking:

In non-blocking, we don't need to wait for first-line output. In this method, if second-line takes less time to first, then our second line will execute first, and after that our first line will execute.

Blocking method executes **synchronously**(line by line) and **non-blocking** execute **asynchronously**.

30. What is Server?

Node.js, often referred to as just Node, is a powerful tool that can run JavaScript applications on both the **server-side** as well as the client-side. **Node.js** can be used to write static file **servers**, Web application frameworks, messaging middleware, and **servers** for HTML5 multiplayer games

31. DOM:

The **DOM** defines a standard for accessing documents: "The W3C Document Object Model (**DOM**) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

32. What is the Class?

A **JavaScript class** is a type of function. Classes are declared with the **class** keyword. We will **use** function expression syntax to initialize a function and **class** expression syntax to initialize a **class**. ... With prototypes, any function can become a constructor instance using the new keyword.

33. What is Map, Filter, Reduce?

➢ **Map**

The **map()** method creates a new array with the results of calling a function for every array element. The **map()** method calls the provided function once for each element in an array, in order. Note: **map()** does not execute the function for array elements without values.

➢ **Filter**

The **filter()** method creates an array filled with all array elements that pass a test (provided as a function). Note: **filter()** does not execute the function for array elements without values.

The `filter()` method **creates a new array** with all elements that pass the test implemented by the provided function.

➢ **Reduce**

The **reduce()** method reduces the array to a single value. ... The return value of the function is stored in an accumulator (result/total). Note: **reduce()** does not execute the function for array elements without values.

34. What is var, let, const:

First, let’s compare **var** and **let**. The main difference between **var** and **let** is that instead of being **function scoped**, let is **block scoped**. What that means is that a variable created with the let keyword is available inside the “block” that it was created in as well as any nested blocks. When I say “block”, I mean anything surrounded by a curly brace {} like in a for loop or an if statement.
The **const** keyword works like the let keyword. But the **const** keyword creates block-scoped variables whose values can't be reassigned.

Simple word:- the main difference between **var**, **let** and **const**, var has **function scope** and let has **block scope** and the const keyword value we **can’t reassign**.

35. What is different between ‘==’ and ‘===’?

difference between "**==**" **and** "**===**" operator **is** that formerly compares variable by making type correction e.g. if you compare a number **with a** string with numeric literal, **==** allows that, but **===** doesn't allow that, because it not only checks the value but also type **of** two variable.

- **” ==” only compares values “===” compare values and type both.**

36. Why we use ‘this’ keyword?

The **JavaScript** this **keyword** refers to the object it belongs to. It has different values depending on where it is used: ... In a function, this refers to the global object. In a function, in strict mode, this is undefined.

37. What is Nan function?

Definition and Usage. The **isNaN**() **function** determines whether a value is an illegal number (Not-a-Number). This **function** returns true if the value equates to **NaN**. Otherwise, it returns false. This **function** is different from the Number specific Number.**isNaN**() method.

38. What is DataType in Javascript?

In **JavaScript,** there are two different kinds of **data**: primitives, and objects. A primitive is simply a **data type** that is not an object and has no methods. In JS, there are six primitive **Data types**:

- Number
- String
- Boolean
- Null
- Undefined
- Symbol

39. What is the difference between Package.json & Package locked.json?

The **package.json** defines the rules for pulling in your dependencies, and the **package.lock.json** describes the exact decencies that were downloaded based on those rules.

40. What is Middleware?

Middleware functions are functions that have access to the request object (req), the response object (res), and the next function in the **application's** request-response cycle. The next function is a function in the Express router which, when invoked, executes the **middleware** succeeding the current **middleware**.

Or

Middleware is a bridge which works between server and client for parsing the data.

Router:

A Router instance is a complete middleware and routing system; for this reason, it is often referred to as a “mini-app”. The following example creates a router as a module, loads a middleware function in it, defines some routes, and mounts the router module on a path in the main app.

41. What is Web hosting?

Web hosting is a service that allows organizations and individuals to post a **website** or **web** page onto the **Internet**. ... Their computer will then connect to your server and your webpages will be delivered to them through the browser. Most **hosting** companies require that you own your **domain** in order to **host** with them.

42. What is Scoping?

Scoping is determining where variables, functions, and objects are accessible in your code during runtime. This means the **scope** of a variable(where it can be accessed) is controlled by the location of the variable declaration. In **Javascript**, there are two types of scopes:-

Scopes:

1. **Global Scope**
   There is only one Global scope throughout a JavaScript document. A variable is in the Global scope if it’s defined outside of a function.
   “Global variables are always defined outside of the function.”
2. **Local Scope**
   Variables declared within a function are in the local scope. Local scope is also called function scope because the local scope is created by functions in Javascript.
   “Local variables are always defined inside of the function.”

- **Scope:**

______Scope defines where variables and functions are accessible inside of your program. In JavaScript, there are two kinds of scope - global scope, and function scope. According to the official spec,

43. What is Http/Https:

1. **Http:**
   HTTP (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide **Web**. As soon as a **Web** user opens their **Web** browser, the user is indirectly making **use** of **HTTP**.

2. **Https:**
   Hypertext Transfer Protocol Secure (**HTTPS**) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network and is widely used on the Internet. ... The protocol is therefore also often referred to as HTTP over TLS, or HTTP over SSL.

44. What is the Event loop?

A programming structure that continuously tests for external events and calls the appropriate routines to handle them. An event loop is often the main loop in a program that usually waits for the user to trigger something.

In the event loop, there are three processes: the 1st one is a stack, 2nd is a web-API, 3rd is a Queue. When we sent multiple requests to server then Server creates an event from our request then it sends into a stack then stack check that event if it's an executable then it executes but if it will take time then its send to web-API to resolve there is a call-back function in which our web- API works when it will resolve then web-API send the data to Queue. Queue works on (First in First out process) in which data it gets first

Only that data will be sent first to Stack. A stack will return to our browser.

1. ***Stack:***
   A stack is a very useful data structure and has a wide range of applications. **The stack** is a linear data structure in which addition or removal of element follows a particular order i.e. LIFO(Last in First Out) AND FILO(First in Last Out).

45. What is Event-Driven Programming?

Event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads. Today, this concept is largely using in GUI Applications wherein a mechanism is in place that listens for events and executes an action once the event occurs. This is the basic principle behind node.js!
OR:
Event-**Driven** programming is a core concept behind **node**.js which is manifested by the implementation of the **Events** module. ... The **event** loop is an entry point used to trigger an **event** that invokes a corresponding **event** handler which in turn can invoke further **events** resulting in the **event driven** programming

46. What is REPL:

REPL stands for ***Read, Eval, Print, Loop*** and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with output in an interactive mode. **Node.js** or **Node** comes bundled with a **REPL** environment.
OR:
REPL stands for ***Read-Eval-Print-Loop***. It is a quick and easy way to test simple Node.js/Javascript to launch the REPL (Node shell) with open terminal.

- **Read:** Reads the user's input, parses it into JavaScript data-structure and then stores it in the memory.
- **Eval:** Receives and evaluates the data structure.
- **Print:** Prints the final result.
- **Loop:** Loops the provided command until *CTRL+C* is pressed twice.

47. What are Cookies?

An HTTP **cookie** is a small piece of data sent from a website and stored on the user's computer by the user's **web browser** while the user is browsing. Cookies were designed to be a reliable mechanism for websites to remember stateful information or to record the user's browsing activity.

48. *What is ORM:*

Object Relational Mapping (**ORM**) is the process of mapping between **objects** and **relational database** systems. So it acts like an interface between two systems hiding details about an underlying mechanism.

49. Sequelize:

Sequelize is a **promise-based ORM** for **Node.js** and io.js. It supports **PostgreSQL**, **MySQL, MariaDB, SQLite** and **MSSQL** and features transaction support, relations, read replication and more. Starting from 4.0.0 Sequelize will only support **Node v4** and above to use **ES6** features.

50. What are Client and Server?:

A client means that **requests** your **resources**. and A server means to **hold some resources.**

51. What is MVC:

MVC is an acronym for **Model-View-Controller**.

It is a design pattern for software projects. It is used majorly by Node developers and by C#, Ruby, PHP framework users too.

a. Model is the data part.
b. View is the User Interface part.
c. Controller is a request-response handler.

52. <u>What is Axios:</u>

Axios is a promise-based HTTP client for the browser and **Node**. **js**. **Axios** makes it easy to send asynchronous HTTP requests to REST endpoints and perform CRUD operations. It can be used in plain **JavaScript** or with a library such as Vue or React.

53. <u>CSV Writer:</u>

CSV Writer. Convert objects/arrays into a **CSV** string or **write** them into a file. It respects RFC 4180 for the **output CSV** format.

54. <u>What is the use of Async await:</u>

The await operator is used to wait for a Promise. It can be used inside an Async block only. The keyword Await makes JavaScript wait until the promise returns a result. It has to be noted that it only makes the async function block wait and not the whole program execution.

55. <u>Morgan</u>:

Morgan is a HTTP request logger middleware for **Node**. **js**. It simplifies the process of logging requests to your **application**.

<u>Database</u>

1. <u>What is Data?</u>

In simple words, data can be facts related to any **object** in **consideration**.
For example, your name, age, height, weight, etc are some data related to you.
A picture , image , file , pdf etc can also be considered data.

2. **What is Database?**

The database is a systematic collection of data. Databases support storage and manipulation of data. Databases make data management easy. Let's discuss a few examples.
Let's also consider **Facebook**. It needs to store, **manipulate** and present data related to members, their friends, member activities, messages, advertisements and a lot more.

Your **electricity** service provider is obviously using a **database** to manage billing, client-related issues, to handle fault data, etc.

3. **What is the Database management system (DBMS)?**

Database Management System (DBMS) is a collection of programs which enables its users to access database, manipulate data, reporting/representation of data.

Some DBMS examples include MySQL, PostgreSQL, Microsoft Access, SQL Server, FileMaker, Oracle, RDBMS, dBASE, Clipper, and FoxPro.

- ★ **Types of DBMS:**

  There are 4 major types of DBMS:-
  - ➔ Hierarchical
  - ➔ Network DBMS
  - ➔ Relational DBMS
  - ➔ Object-Oriented Relation DBMS

4. **What is Relational database management system (RDBMS)?**

RDBMS. Stands for "**Relational Database Management System**." An **RDBMS** is a **DBMS** designed specifically for **relational databases**. ... A **relational database** refers to a **database** that stores data in a structured format, using rows and columns. This makes it easy to locate and access specific values within the **database**.

5. **What is SQL:**

______Structured Query Language (SQL) is a standard computer language for relational database management and data manipulation. SQL is used to query, insert, update and modify data.

6. **What is MySQL:**

MySQL is a relational **database** management system based on SQL – Structured Query Language. The application is **used for** a wide range of purposes, including data warehousing, e-commerce, and logging applications. The most common use for **mySQL** however, is for the purpose of a web **database**