Gauthamjm007 / **Backend-NodeJS-Golang-Interview_QA** Public

A collection of Node JS and Golang Backend interview questions please feel free to fork and contribute to this repository

MIT license

626 stars **159** forks Branches Tags Activity

Star Notifications

Code Issues 7 Pull requests Actions Projects Security Insights

master **3** Branches **0** Tags Go to file Go to file Code

Gauthamjm007 adding new question to database engineering 2cbec61 · 4 years ago

img	adding three q and a	4 years ago
.prettierignore	updating space in backto	4 years ago
LICENSE	Create LICENSE	5 years ago
README.md	adding new question to database engineering	4 years ago
sentence_to_hashing_converter.js	adding new question to database engineering	4 years ago
test.txt	addign 28 TO 30th Q	4 years ago

Backend Interview Questions and Answers related to Node JS , Golang/Go , Express JS , MongoDB

> Click ⭐ if you like the project. Pull Request are highly appreciated.
> Node JS
> Express JS
> MongoDB and Mongoose
> Database Engineering
> Golang

Table of Contents - Node JS

No.	Questions
14	List down the two arguments that async queue takes as input?
15	Differentiate between spawn and fork methods in Nodejs?
16	Explain the purpose of ExpressJS package?
17	Explain the usage of a buffer class in Nodejs?
18	How does Nodejs handle the child threads?
19	Explain stream in Nodejs along with its various types?
20	Describe the exit codes of Nodejs?
21	Is cryptography supported in Nodejs?
22	Explain the reason as to why Express app and server folder must be kept separate?
23	What is the role of asset module in nodejs?
24	What is the role of async_hooks module in nodejs?
25	What are buffer objects in nodejs?
26	What are the different ways of implementing Addons in NodeJS?
27	How can we spawn the child process asynchronously without blocking the Nodejs event loop?
28	How can we take advantage of multi-core system in Nodejs as nodejs works on single thread?
29	What is the datatype of console?
30	Which are the different console methods available?
31	Can node js perform cryptographic functions?
32	How can we read or write files in node js?
33	Which are the global objects in Node JS?
34	How can we perform asynchronous network API in Node JS?
35	What are the utilities of OS module in NodeJS?
36	Which are the areas where it is suitable to use NodeJS?
37	Which are the areas where it is not suitable to use NodeJS?
38	What Are The Key Features Of NodeJs?
39	Explain REPL In NodeJs?
40	Can you write CRUD operations in Node js without using frameworks?
41	Can You Create HTTP Server In Nodejs Explain The Code Used For It?
42	What Is The Difference Between Nodejs AJAX And JQuery?
43	What Is EventEmitter In NodeJs?
44	What Is A Child_process Module In NodeJs?

Node Js

1. ### What is NodeJS?

   Node.js is an open-source, cross-platform, **JavaScript runtime environment that executes JavaScript code outside of a browser**. Node JS was created by Ryan Dahl, Ryan Dahl is a software engineer and the original developer of the Node.js JavaScript runtime.

Back to Top ⬆

2. ### How can you avoid callback hells?

There are lots of ways to solve the issue of callback hells:
1.modularization: break callbacks into independent functions,
2.use a control flow library, like async.
3.use generators with Promises,
4.use async/await (note that it is only available in the latest v7 release and not in the LTS version

Back to Top ⬆

3. **When are background or worker processes useful?**

   Worker processes are extremely useful if you'd like to do data processing in the background, like sending out emails or processing images.
   There are lots of options for this like RabbitMQ or Kafka.

Back to Top ⬆

4. **Why is NodeJS Single threaded?**

   Node.js is single-threaded for async processing. By doing async processing on a single-thread under typical web loads, more performance and scalability can be achieved as opposed to the typical thread-based implementation.

Back to Top ⬆

5. **Name the types of API functions in Node?**

   There are two types of functions in Node.js.
   1.Blocking functions - In a blocking operation, all other code is blocked from executing until an I/O event that is being waited on occurs. Blocking functions execute synchronously.
   2.Non-blocking functions - In a non-blocking operation, multiple I/O calls can be performed without the execution of the program being halted. Non-blocking functions execute asynchronously.

Back to Top ⬆

6. **Explain chaining in Nodejs?**

Back to Top ⬆

7. **What are streams in Nodejs Explain the different types of streams present in Nodejs?**

   Streams are objects that allow the reading of data from the source and writing of data to the destination as a continuous process.
   There are four types of streams.
   to facilitate the reading operation.
   to facilitate the writing operation.
   to facilitate both read and write operations.
   is a form of Duplex stream that performs computations based on the available input.

Back to Top ⬆

8. **What is package json?**

   The package.json file in Node.js is the heart of the entire application. It is basically the manifest file that contains the metadata of the project where we define the properties of a package.

Back to Top ⬆

9. **Explain the purpose of module exports?**

   A module in Node.js is used to encapsulate all the related codes into a single unit of code which can be interpreted by shifting all related functions into a single file

Back to Top ⬆

10. **List down the major security implementations within Nodejs?**

    Major security implementations in Node.js are:Authentications,Error Handling

Back to Top ⬆

11. **Explain the concept of URL module?**

    The URL module splits up a web address into readable parts

Back to Top ⬆

12. **Explain the concept of middleware in Nodejs?**

    In general, middleware is a function receives the Request and Response objects. In other words, in an application's request-response cycle these functions have access to various request & response objects along with the next function of the cycle,The next function of middleware is represented with the help of a variable, usually named next. Most commonly performed tasks by the middleware functions are

    Execute any type of code
    Update or modify the request and the response objects
    Finish the request-response cycle
    Invoke the next middleware in the stack

Back to Top ⬆

13. **Explain libuv?**

    Libuv is a multi-platform support library of Node.js which majorly is used for asynchronous I/O. It was primarily developed for Node.js, with time it is popularly practiced with other systems like as Luvit, pyuv, Julia, etc. Libuv is basically an abstraction around libev/ IOCP depending on the platform, providing users an API based on libev. A few of the important features of libuv are:

    Full-featured event loop backed
    File system events
    Asynchronous file & file system operations
    Asynchronous TCP & UDP sockets
    Child processes

Back to Top ⬆

14. **List down the two arguments that async.queue takes as input?**

    Below are the two arguments that async.queue takes as input - Task Function & Concurrency Value

Back to Top ⬆

15. **Differentiate between spawn and fork methods in Nodejs?**

    In Node.js, the spawn() is used to launch a new process with the provided set of commands. This method doesn't create a new V8 instance and just one copy of the node module is active on the processor. When your child process returns a large amount of data to the Node you can invoke this method.

Back to Top ⬆

16. **Explain the purpose of ExpressJS package?**

    Express.js is a framework built on top of Node.js that facilitates the management of the flow of data between server and routes in the server-side applications. It is a lightweight and flexible framework that provides a wide range of features required for the web as well as mobile application development. Express.js is developed on the middleware module of Node.js called connect. The connect module further makes use of http module to communicate with Node.js. Thus, if you are working with any of the connect based middleware modules, then you can easily integrate with Express.js.

Back to Top ⬆

17. **Explain the usage of a buffer class in Nodejs?**

    Buffer class in Node.js is used for storing the raw data in a similar manner of an array of integers. But it corresponds to a raw memory allocation that is located outside the V8 heap. It is a global class that is easily accessible can be accessed in an application without importing a buffer module. Buffer class is used because pure JavaScript is not compatible with binary data. So, when dealing with TCP streams or the file system, it's necessary to handle octet streams.

Back to Top ⬆

18. **How does Nodejs handle the child threads?**

In general, Node.js is a single threaded process and doesn't expose the child threads or thread management methods. But you can still make use of the child threads using spawn() for some specific asynchronous I/O tasks which execute in the background and don't usually execute any JS code or hinder with the main event loop in the application. If you still want to use the threading concept in your application you have to include a module called ChildProcess explicitly.

Back to Top ⬆

19. **Explain stream in Nodejs along with its various types?**

Streams in Node.js are the collection of data similar to arrays and strings. They are objects using which you can read data from a source or write data to a destination in a continuous manner. It might not be available at once and need not to have fit in the memory. These streams are especially useful for reading and processing a large set of data. In Node.js, there are four fundamental types of streams:
Readable: Used for reading large chunks of data from the source.
Writeable: Use for writing large chunks of data to the destination.
Duplex: Used for both the functions; read and write.
Transform: It is a duplex stream that is used for modifying the data.

Back to Top ⬆

20. **Describe the exit codes of Nodejs?**

In Node.js, exit codes are a set of specific codes which are used for finishing a specific process. These processes can include the global object as well. Below are some of the exit codes used in Node.js:
*Uncaught fatal exception
*Unused
*Fatal Error
*Internal Exception handler Run-time failure
*Internal JavaScript Evaluation Failure

Back to Top ⬆

21. **Is cryptography supported in Nodejs?**

Yes, Node.js does support cryptography through a module called Crypto. This module provides various cryptographic functionalities like cipher, decipher, sign and verify functions, a set of wrappers for open SSL's hash HMAC etc.

```
const crypto = require'crypto');
const secret = 'akerude';
const hash = crypto.createHmac('swaEdu', secret).update('Welcome to Edureka').digest('hex');
console.log(hash);
```

Back to Top ⬆

22. **Explain the reason as to why Express app and server folder must be kept separate?**

Express 'app' and 'server' must be kept separate as by doing this, you will be separating the API declaration from the network related configuration which benefits in the below listed ways:
*It allows testing the API in-process without having to perform the network calls
*Faster testing execution
*Getting wider coverage metrics of the code
*Allows deploying the same API under flexible and different network conditions
*Better separation of concerns and cleaner code

Back to Top ⬆

23. **What is the role of asset module in nodejs?**

The assert module provides a set of assertion functions for verifying invariants

Back to Top ⬆

24. **What is the role of async_hooks module in nodejs?**

The async_hooks module provides an API to track asynchronous resources. It can be accessed using:

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

Back to Top ⬆

25. ### What are buffer objects in nodejs?

In Node.js, Buffer objects are used to represent binary data in the form of a sequence of bytes. Many Node.js APIs, for example streams and file system operations, support Buffers, as interactions with the operating system or other processes generally always happen in terms of binary data

Back to Top ⬆

26. ### What are the different ways of implementing Addons in NodeJS?

There are three options for implementing Addons:
N-API
nan direct use of internal V8
libuv
Node.js libraries

Back to Top ⬆

27. ### How can we spawn the child process asynchronously without blocking the Nodejs event loop?

child_process.spawn() method spawns the child process asynchronously, without blocking the Node.js event loop,The child_process. **spawnSync()** function provides equivalent functionality in a synchronous manner that blocks the event loop until the spawned process either exits or is terminated

Back to Top ⬆

28. ### How can we take advantage of multi-core system in Nodejs as nodejs works on single thread?

We can use node js cluster to use multicores in the hardware,The cluster module allows easy creation of child processes that all share server ports

```
const cluster = require('cluster');
const http = require('http');
const numCPUs = require('os').cpus().length;

if (cluster.isMaster) {
console.log(`Master ${process.pid} is running`);

// Fork workers.
    for (let i = 0; i < numCPUs; i++) {
cluster.fork();
}

cluster.on('exit', (worker, code, signal) => {
console.log(`worker ${worker.process.pid} died`);
    });
}
else {
    // Workers can share any TCP connection
    // In this case it is an HTTP server
http.createServer((req, res) => {
res.writeHead(200);
res.end('hello world\n');
}).listen(8000);

console.log(`Worker ${process.pid} started`);
}

//Running Node.js will now share port 8000 between the workers:

$ node server.js
Master 3596 is running
Worker 4324 started
Worker 4520 started
Worker 6056 started
Worker 5644 started
```

Back to Top ⬆

29. ### What is the datatype of console?

The datatype of console is an **object**

Back to Top ⬆

30. Which are the different console methods available?

There are around 21 inbuilt console methods , we can also built our own prototypes using new Console constructor function
here are a few popular one's
1.**console.clear()** will clear only the output in the current terminal viewport for the Node.js binary.
2.**console.error([data][, ...args])** Prints to stderr with newline. Multiple arguments can be passed, with the first used as the primary message and all additional used as substitution
3.**console.table(tabularData[, properties])** a table with the columns of the properties of tabularData (or use properties) and rows of tabularData and log it.

Back to Top ⬆

31. Can node js perform cryptographic functions?

Yes,The crypto module provides cryptographic functionality that includes a set of wrappers for OpenSSL's hash, HMAC, cipher, decipher, sign, and verify functions.

Use require('crypto') to access this module.

Back to Top ⬆

32. How can we read or write files in node js?

The fs module provides an API for interacting with the file system in a manner closely modeled around standard POSIX functions. To use this module:

const fs = require('fs');
There are a few methods like
fs.readFile(file, data[, options], callback)
fs.writeFile(file, data[, options], callback)

Back to Top ⬆

33. Which are the global objects in Node JS?

__dirname
__filename
clearImmediate(immediateObject)
clearInterval(intervalObject)
clearTimeout(timeoutObject)
console
exports
global
module
process
queueMicrotask(callback)
require()
setImmediate(callback[, ...args])
setInterval(callback, delay[, ...args])
setTimeout(callback, delay[, ...args])
TextDecoder
TextEncoder
URL
URLSearchParams
WebAssembly

Back to Top ⬆

34. How can we perform asynchronous network API in Node JS?

The net module provides an asynchronous network API for creating stream-based TCP or IPC servers (net.createServer()) and clients (net.createConnection()).

It can be accessed using:

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

Back to Top ⬆

35. ### What are the utilities of OS module in NodeJS?

    The os module provides operating system-related utility methods and properties. It can be accessed using:

    const os = require('os');.

Back to Top ⬆

36. ### Which are the areas where it is suitable to use NodeJS?

    I/O bound Applications
    Data Streaming Applications
    Data Intensive Real-time Applications (DIRT)
    JSON APIs based Applications
    Single Page Applications
    .

Back to Top ⬆

37. ### Which are the areas where it is not suitable to use NodeJS?

    it's not suitable for heavy applications involving more of CPU usage

Back to Top ⬆

38. ### What Are The Key Features Of NodeJs?

    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.
    .

Back to Top ⬆

39. ### Explain REPL In NodeJs?

    The REPL stands for "Read Eval Print Loop". It is a simple program that accepts the commands, evaluates them, and finally prints the results. REPL provides an environment similar to that of Unix/Linux shell or a window console, in which we can enter the command and the system, in turn, responds with the output. REPL performs the following tasks.

    READ - It Reads the input from the user, parses it into JavaScript data structure and then stores it in the memory.
    EVAL - It Executes the data structure.
    PRINT - It Prints the result obtained after evaluating the command.
    LOOP - It Loops the above command until the user presses Ctrl+C two times.

.

Back to Top ⬆ ?

40. ### Can you write CRUD operations in Node js without using frameworks?

    Yes,we can use inbuilt http library for that , here is a simple code for the same:

```
var http = require('http');//create a server object:
http.createServer(function (req, res) {
res.writeHead(200, {'Content-Type': 'text/html'}); // http headervar    url = req.url;
if(url ==='/about'){
    res.write('<h1>about us page<h1>'); //write a response
    res.end(); //end the response
}else if(url ==='/contact'){
    res.write('<h1>contact us page<h1>'); //write a response
    res.end(); //end the response
}else{
    res.write('<h1>Hello World!<h1>'); //write a response
    res.end(); //end the response
}}).listen(3000, function(){
console.log("server start at port 3000"); //the server object listens on port 3000
});
```

.

Back to Top ⬆

42. What Is The Difference Between Nodejs AJAX And JQuery?

The one common trait between Node.js, AJAX, and jQuery is that all of them are the advanced implementation of JavaScript. However, they serve completely different purposes.

Node.Js :

It is a server-side platform for developing client-server applications. For example, if we've to build an online employee management system, then we won't do it using client-side JS. But the Node.js can certainly do it as it runs on a server similar to Apache, Django not in a browser.

AJAX (Aka Asynchronous Javascript And XML) :

It is a client-side scripting technique, primarily designed for rendering the contents of a page without refreshing it. There are a no. of large companies utilizing AJAX such as Facebook and Stack Overflow to display dynamic content.

JQuery :

It is a famous JavaScript module which complements AJAX, DOM traversal, looping and so on. This library provides many useful functions to help in JavaScript development. However, it's not mandatory to use it but as it also manages cross-browser compatibility, so can help you produce highly maintainable web applications.

.

Back to Top ⬆

43. What Is EventEmitter In NodeJs?

Events module in Node.js allows us to create and handle custom events. The Event module contains "EventEmitter" class which can be used to raise and handle custom events. It is accessible via the following code.

```
// Import events module
var events = require('events');

// Create an eventEmitter object
var eventEmitter = new events.EventEmitter();
```

When an EventEmitter instance encounters an error, it emits an "error" event. When a new listener gets added, it fires a "newListener" event and when a listener gets removed, it fires a "removeListener" event.

EventEmitter provides multiple properties like "on" and "emit". The "on" property is used to bind a function to the event and "emit" is used to fire an event. .

Back to Top ⬆

44. What Is A Child_process Module In NodeJs?

Node.js supports the creation of child processes to help in parallel processing along with the event-driven model.

The Child processes always have three streams <child.stdin>, child.stdout, and child.stderr. The stream of the parent process shares the streams of the child process.

Node.js provides a <child_process> module which supports following three methods to create a child process.

```
**exec** – <child_process.exec> method runs a command in a shell/console and buffers the output.
**spawn** – <child_process.spawn> launches a new process with a given command.
**fork** – <child_process.fork> is a special case of the spawn() method to create child processes.
```

.

Back to Top ⬆

Table of Contents - Express JS

Express Js

1. ### What is ExpressJS?

Fast, unopinionated, minimalist web framework for Node.js,Express is a project of the Node.js Foundation.it is open source and click here to view,it is mainly used for building networking services (web applications) and applications,it is builds on top of node js features to provide easy to use functionality that satisfy the needs for building web applications.it has lots of pre-built packages and also many framework are built on top of it like
Feathers: Build prototypes in minutes and production ready real-time apps in days
NestJs: A progressive Node.js framework for building efficient, scalable, and enterprise-grade server-side applications on top of TypeScript & JavaScript (ES6, ES7, ES8)
Sails: MVC framework for Node.js for building practical, production-ready apps.

Back to Top ⬆

2. ### What are some of the salient features of express?

Middlewares: Set up middlewares in order to respond to HTTP/RESTful Requests.
Routing: It is possible to defines a routing table in order to perform different HTTP operations.
Templates: Dynamically renders HTML Pages based on passing arguments to templates.
High Performance: Express prepare a thin layer, therefore, the performance is adequate.
Database Support: Express supports RDBMS as well as NoSQL databases.
MVC Support: Organize the web application into an MVC architecture. **Manages everything from routes** to rendering view and preforming HTTP request.

Back to Top ⬆

3. ### Explain with an example a working of a simple express app?

I have given you the code, you explain , don't expect me to do all the stuff

```
const express = require('express')
const port = 3000
const app = express()
app.get('/', function(req, res) {
  res.send('Hello World!')
})
app.listen(port, function(){
  console.log('listening on port',port)
})
```

Back to Top ⬆

4. ### Mention few properties of request parameter in express?

here is a list of few req methods needed for you to knows

Property	Description
.app	holds a reference to the Express app object
.baseUrl	the base path on which the app responds
.body	contains the data submitted in the request body (must be parsed and populated manually before you can access it)
.cookies	contains the cookies sent by the request (needs the `cookie-parser` middleware)
.hostname	the server hostname
.ip	the server IP
.method	the HTTP method used
.params	the route named parameters
.path	the URL path
.protocol	the request protocol
.query	an object containing all the query strings used in the request
.secure	true if the request is secure (uses HTTPS)
.signedCookies	contains the signed cookies sent by the request (needs the `cookie-parser` middleware)
.xhr	true if the request is an XMLHttpRequest

Back to Top ⬆

5. ### How to get the name parameters in express?

This property is an object containing properties mapped to the named route “parameters”. For example, if you have the route /user/:name, then the “name” property is available as req.params.name. This object defaults to {}.

```
// GET /user/tj
req.params.name
// => "tj"
```

Back to Top ⬆

6. **How to retrieve the get query string parameters using express?**

The query string is the part that comes after the URL path, and starts with a question mark ?.

```
?height=6&weight=60
//req.query.height - 6
//req.query.weight - 60
```

Back to Top ⬆

7. **How to send a response back using express?**

we can use any one of these commands

```
function(req, res) {
    res.send('Hello World!')
}
function(req, res) {
    res.end('Hello World!')
}
function(req, res) {
    res.json({title:'Hello World!'})
}
```

Back to Top ⬆

8. **How to set http response status using express?**

we can either use **res.status()** or **res.sendStatus()**

```
res.status(404).send('File not found')

//if sendStatus we no need to write send method , i will pre send a few inbuilt messages upon using that

res.sendStatus(200)
// === res.status(200).send('OK')

res.sendStatus(403)
// === res.status(403).send('Forbidden')

res.sendStatus(404)
// === res.status(404).send('Not Found')

res.sendStatus(500)
// === res.status(500).send('Internal Server Error')
```

Back to Top ⬆

9. **What are the different http status codes?**

Code	Message	Description
100	Continue	Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request
101	Switching Protocols	The server switches protocol
200	OK	The request is OK.
201	Created	The request is complete, and a new resource is created
202	Accepted	The request is accepted for processing, but the processing is not complete
203	Non-authoritative Information	The information in the entity header is from a local or third-party copy, not from the original server.
204	No Content	A status code and a header are given in the response but there is no entity-body in the reply
205	Reset Content	The browser should clear the form used for this transaction for additional input
206	Partial Content	The server is returning partial data of the size requested. Used in response to a request specifying a Range header. The server must specify the range included in the response with the Content-Range header

Code	Message	Description
300	Multiple Choices	A link list. The user can select a link and go to that location. Maximum five addresses
301	Moved Permanently	The requested page has moved to a new url
302	Found	The requested page has moved temporarily to a new url
303	See Other	The requested page can be found under a different url
304	Not Modified	This is the response code to an If-Modified-Since or If-None-Match header, where the URL has not been modified since the specified date
305	Use Proxy	The requested URL must be accessed through the proxy mentioned in the Location header
306	Unused	This code was used in a previous version. It is no longer used, but the code is reserved
307	Temporary Redirect	The requested page has moved temporarily to a new url
400	Bad Request	The server did not understand the request
401	Unauthorized	The requested page needs a username and a password
402	Payment Required	You can not use this code yet.
403	Forbidden	Access is forbidden to the requested page.
404	Not Found	The server can not find the requested page.
405	Method Not Allowed	The method specified in the request is not allowed
406	Not Acceptable	The server can only generate a response that is not accepted by the client
407	Proxy Authentication Required	You must authenticate with a proxy server before this request can be served
408	Request Timeout	The request took longer than the server was prepared to wait
409	Conflict	The request could not be completed because of a conflict
410	Gone	The requested page is no longer available
411	Length Required	The "Content-Length" is not defined. The server will not accept the request without it .
412	Precondition Failed	The pre condition given in the request evaluated to false by the server
413	Request Entity Too Large	The server will not accept the request, because the request entity is too large.
414	Request-url Too Long	The server will not accept the request, because the url is too long. Occurs when you convert a "post" request to a "get" request with a long query information
415	Unsupported Media Type	The server will not accept the request, because the mediatype is not supported
416	Requested Range Not Satisfiable	The requested byte range is not available and is out of bounds
417	Expectation Failed	The expectation given in an Expect request-header field could not be met by this server.
500	Internal Server Error	The request was not completed. The server met an unexpected condition.
501	Not Implemented	The request was not completed. The server did not support the functionality required.
502	Bad Gateway	The request was not completed. The server received an invalid response from the upstream server.
503	Service Unavailable	The request was not completed. The server is temporarily overloading or down.
504	Gateway Timeout	The gateway has timed out.
505	HTTP Version Not Supported	The server does not support the "http protocol" version

Back to Top ⬆

10. **Mention few properties of request parameter in express?**

You can access all the HTTP headers using the Request.headers property:

```
app.get('/', (req, res) => {
console.log(req.headers)
})

app.get('/', (req, res) => {
req.header('User-Agent')
})
```

Back to Top ⬆

11. **How can you change http header value of a response?**

You can change any HTTP header value using Response.set():

```
res.set('Content-Type', 'text/html')
res.type('json')
// => 'application/json'

res.type('application/json')
// => 'application/json'

res.type('png')
// => image/png:
```

Back to Top ⬆

12. **How to redirect to other pages server-side?**

Redirects are common in Web Development. You can create a redirect using the Response.redirect() method:

```
res.redirect('/go-there')
//it can be either a url or a path of file
res.redirect(301, '/go-there')
```

Back to Top ⬆

13. **How does routing work in express?**

Routing is the process of determining what should happen when a URL is called, or also which parts of the application should handle a specific incoming request.

In the Hello World example we used this code

```
app.get('/', function(req, res) {
 /* */
})
//This creates a route that maps accessing the root domain URL / using the HTTP GET method to the response we
want to provide.
```

Back to Top ⬆

14. **What are the tasks that a middleware can do?**

Middleware functions can perform the following tasks:

Execute any code.
Make changes to the request and the response objects.
End the request-response cycle.
Call the next middleware function in the stack.

Back to Top ⬆

15. **What are the different types of middleware?**

An Express application can use the following types of middleware:

Application-level middleware
Router-level middleware
Error-handling middleware
Built-in middleware
Third-party middleware

Back to Top ⬆

16. ## How to serve static assests from express?

It's common to have images, CSS and more in a public subfolder, and expose them to the root level:

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

app.use(express.static('public'))

app.listen(3000, () => console.log('Server ready'))
```

Back to Top ⬆

17. ## How to provide file download using express?

Express provides a handy method to transfer a file as attachment: Response.download().

Once a user hits a route that sends a file using this method, browsers will prompt the user for download.

The Response.download() method allows you to send a file attached to the request, and the browser instead of showing it in the page, it will save it to disk.

```
app.get('/', (req, res) => res.download('./file.pdf'))
```

Back to Top ⬆

18. ## How to use the Response.cookie() method to manipulate your cookies?

Cookies are small pieces of data sent from a website and are stored in user's web browser while user is browsing that website. Every time the user loads that website back, the browser sends that stored data back to website or server, to distinguish user's previous activity.

```
res.cookie('username', 'Adam')

This method accepts a third parameter which contains various options:
res.cookie('username', 'Adam', { domain: '.bangalore.com', path: '/administrator', secure: true })

res.cookie('username', 'Adam', { expires: new Date(Date.now() + 900000), httpOnly: true })

//clear cookie
res.clearCookie('username')
```

The most useful parameters you can set are:

Value	Description
domain	the cookie domain name
expires	set the cookie expiration date. If missing, or 0, the cookie is a session cookie
httpOnly	set the cookie to be accessible only by the web server. See HttpOnly
maxAge	set the expiry time relative to the current time, expressed in milliseconds
path	the cookie path. Defaults to /
secure	Marks the cookie HTTPS only
signed	set the cookie to be signed
sameSite	Value of SameSite

Back to Top ⬆

19. ### How to manage sessions using express?

We'll use the express-session module, which is maintained by the Express team.When implemented, every user of you API or website will be assigned a unique session, and this allows you to store the userstate.as by default Express requests are sequential and no request can be linked to each other. There is no way to know if this request comes from a client that already performed a request previously.

```
const express = require('express')
const session = require('express-session')

const app = express()
app.use(session(
'secret': '343ji43j4n3jn4jk3n'
))
```

All solutions store the session id in a cookie, and keep the data server-side. The client will receive the session id in a cookie, and will send it along with every HTTP request.

We'll reference that server-side to associate the session id with the data stored locally.

Memory is the default, it requires no special setup on your part, it's the simplest thing but it's meant only for development purposes.

The best choice is a memory cache like Redis, for which you need to setup its own infrastructure.

Back to Top ⬆

20. ### How to process forms using Express?

The form data will be sent in the POST request body.

To extract it, you will use the express.urlencoded() middleware, provided by Express:

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

app.use(express.urlencoded())
```

Now you need to create a POST endpoint on the /submit-form route, and any data will be available on Request.body:

```
app.post('/submit-form', (req, res) => {
const username = req.body.username
//...
res.end()
})
```

Back to Top ⬆

21. ### How To Allow Cors In Expressjs Explain With An Example?

In order to allow CORS in Express.js, add the following code in server.js:

```
app.all('*', function(req, res, next) {
res.set('Access-Control-Allow-Origin', '*');
res.set('Access-Control-Allow-Methods', 'GET, POST, DELETE, PUT');
res.set('Access-Control-Allow-Headers', 'X-Requested-With, Content-Type');
if ('OPTIONS' == req.method) return res.send(200);
next();
});
```

or you can install a package called cors ,CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.link

```
var express = require('express')
var cors = require('cors')
var app = express()

app.use(cors())
```

Back to Top ⬆

Table of Contents - MongoDB and Mongoose

1. What is MongoDB?

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability.Classified as a NoSQL database program, MongoDB uses JSON-like documents with schema.
MongoDB is written in C++

MongoDB is known to be used by the City of Chicago, Codecademy, Google Search, Foursquare, IBM, Orange S.A., The Gap, Inc., Uber, Coinbase, Sega, Barclays, HSBC, eBay, Cisco, Bosch and Urban Outfitters

Back to Top ⬆

2. What are the difference between NoSQL and SQL?

Parameter	SQL	NOSQL
Definition	SQL databases are primarily called RDBMS or Relational Databases	NoSQL databases are primarily called as Non-relational or distributed database
Query Language	Structured query language (SQL)	No declarative query language
Type	SQL databases are table based databases	NoSQL databases can be document based, key-value pairs, graph databases
Schema	SQL databases have a predefined schema	NoSQL databases use dynamic schema for unstructured data.
Ability to scale	SQL databases are vertically scalable	NoSQL databases are horizontally scalable
Examples	Oracle, Postgres, and MS-SQL.	MongoDB, Redis, Neo4j, Cassandra, Hbase.
Best suited for	An ideal choice for the complex query intensive environment.	It is not good fit complex queries.
Hierarchical data storage	SQL databases are not suitable for hierarchical data storage.	More suitable for the hierarchical data store as it supports key-value pair method
Variations	One type with minor variations	Many different types which include key-value stores, document databases, and graph databases

Parameter	SQL	NOSQL
Consistency	It should be configured for strong consistency.	It depends on DBMS as some offers strong consistency like MongoDB, whereas others offer only offers eventual consistency, like Cassandra.
Hardware	Specialized DB hardware (Oracle Exadata, etc.)	Commodity hardware
Network	Highly available network (Infiniband, Fabric Path, etc.)	Commodity network (Ethernet, etc.)
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool.
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
ACID vs. BASE Mode	ACID(Atomicity, Consistency, Isolation, and Durability) is a standard for RDBMS	Base (Basically Available, Soft state, Eventually Consistent) is a model of many NoSQL systems
Average salary	₹ 5,58,704 per year	₹ 6,04,959 per year

Back to Top ⬆

3. ## How to establish MongoDB database connection in a node application?

Database Connection Create a file ./config/database.js under the project root.

Database Connection

Next, we will add code that connects to the database.

in database.js file

```
const mongoose = require('mongoose');
const server = '127.0.0.1:27017'; // REPLACE WITH YOUR DB SERVER
const database = 'fcc-Mail';      // REPLACE WITH YOUR DB NAME

 mongoose.connect(`mongodb://${server}/${database}`)
   .then(() => {
     console.log('Database connection successful')
   })
   .catch(err => {
     console.error('Database connection error')
   })

module.exports = { mongoose }
```

MongoDB Atlas sign up to mongosb atlas and it will help you make a connection by url, having a secret key and password

Back to Top ⬆

4. ## What are virtual property in mongoose?

A virtual property is not persisted to the database. We can add it to our schema as a helper to get and set values.but it wont store in database

```
userSchema.virtual('fullName').get(function() {
return this.firstName + ' ' + this.lastName
})

userSchema.virtual('fullName').set(function(name) {
let str = name.split(' ')

this.firstName = str[0]
this.lastName = str[1]
})

const user = new User()
user.fullName = 'Thomas Anderson'
console.log(user.toJSON())  // Output model fields as JSON
console.log()
console.log(user.fullName)  // Output the full name
//The code above will output the following:

{ _id: 5a7a4248550ebb9fafd898cf,
```

```
  firstName: 'Thomas',
  lastName: 'Anderson' }
//Thomas Anderson
```

Back to Top ⬆

5. **How can we add or create our own instance methods in mongoose?**

   We can create custom helper methods on the schema and access them via the model instance. These methods will have access to the model object and they can be used quite creatively

```
userSchema.methods.details = function() {
return this.username + ' - ' +  this.email
}
//This method will be accessible via a model instance:
const user = new User({
username: 'user2',
email: 'user2@gmail.com'
})
```

Back to Top ⬆

6. **How can we add or create our own static methods in mongoose?**

Similar to instance methods, we can create static methods on the schema. Let's create a method to retrieve all users in the database:

```
userSchema.statics.getUsers = function() {
return new Promise((resolve, reject) => {
  this.find((err, docs) => {
    if(err) {
      console.error(err)
      return reject(err)
    }
    resolve(docs)
        })
    })
  }
```

Back to Top ⬆

7. **What are the mongoose middlewares?**

   Middleware are functions that run at specific stages of a pipeline. Mongoose supports middleware for the following operations:

   Aggregate
   Document
   Model
   Query

Back to Top ⬆

8. **How to query data using mongoose?**

   Mongoose has a very rich API that handles many complex operations supported by MongoDB. Consider a query where we can incrementally build query components.

   In this example, we are going to:

   Find all users
   Skip the first 100 records
   Limit the results to 10 records
   Sort the results by the firstName field
   Select the firstName
   Execute that query

```
  User.find()                  // find all users
       .skip(100)               // skip the first 100 items
       .limit(10)               // limit to 10 items
       .sort({firstName: 1}      // sort ascending by firstName
       .select({firstName: true} // select firstName only
```

```
.exec()                  // execute the query
.then(docs => {
   console.log(docs)
 })
.catch(err => {
   console.error(err)
 })
```

Back to Top ⬆

9. What is Population in mongoose?

Population is the process of automatically replacing the specified paths in the document with document(s) from other collection(s). We may populate a single document, multiple documents, plain object, multiple plain objects, or all objects returned from a query.

Back to Top ⬆

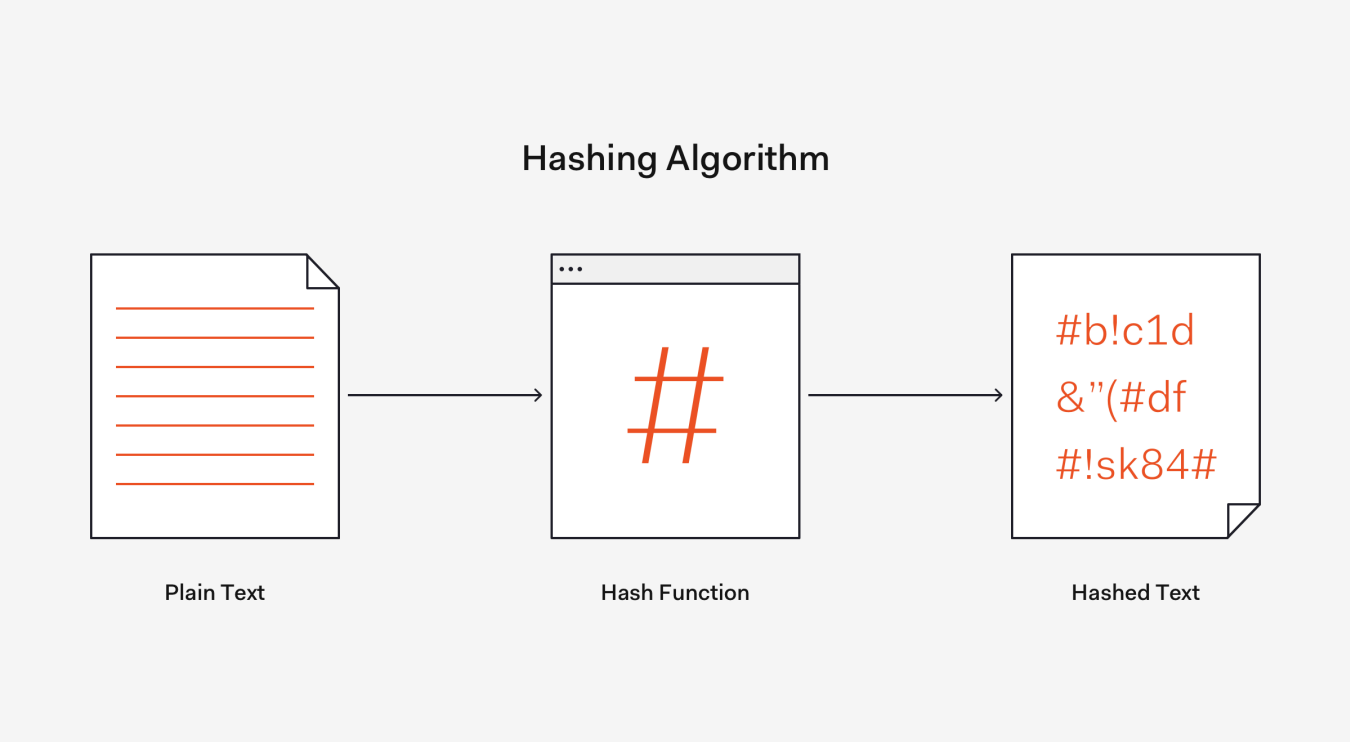
10. What is Datamasking?

Data masking is a method of creating a structurally similar but inauthentic version of an organization's data that can be used for purposes such as software testing and user training. The purpose is to protect the actual data while having a functional substitute for occasions when the real data is not required.
you can simply use **$project** to hide the mobile field
Or perhaps you have an extra field in your document to indicate whether the information is public or not, i.e. given documents

Back to Top ⬆

11. What is hashing and explain how it works?

Hashing is the process of converting an input of any length into a fixed size string of text, using a mathematical function.



When the user provides a input it will be converted to a value of fixed length by a hashing function and the resulting value will be called as hashed text, and it should be always unique for different value

Back to Top ⬆

12. What are salts and why are they so important?

It's a unique value that can be added to the end of the password to create a different hash value. This adds a layer of security to the hashing process
They are so important as they prevent **brute force attacks**(Trying all possible combintaion of password) and also against **rainbow table**(a table containing all common hashed text and their respective passwords)

Back to Top ⬆

13. What are pepper and why are they so important?

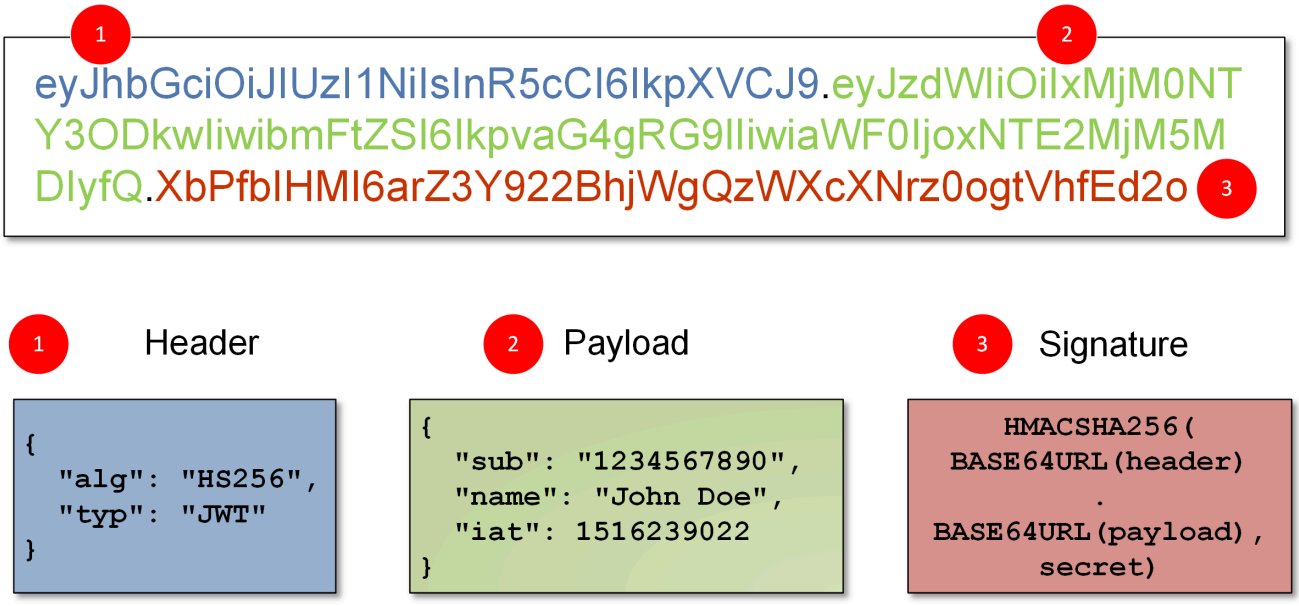
A pepper is a secret added to an input such as a password prior to being hashed with a cryptographic hash function
A pepper performs a comparable role to a salt, but while a salt is not secret (merely unique) and can be stored alongside the hashed output
A pepper is secret and must not be stored with the output. The hash and salt are usually stored in a database, but a pepper must be stored separately (e.g. in a configuration file) to prevent it from being obtained by the attacker in case of a database breach.
Where the salt only has to be long enough to be unique, a pepper has to be secure to remain secret (at least 112 bits is recommended by NIST), otherwise an attacker only needs one known entry to crack the pepper.
Finally, the pepper must be generated anew for every application it is deployed in, otherwise a breach of one application would result in lowered security of another application.

Back to Top ⬆

14. **What are JWT?**

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object
some scenarios where JSON Web Tokens are useful:
Authorization: This is the most common scenario for using JWT. Once the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains.

Information Exchange: JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed—for example, using public/private key pairs—you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.



Back to Top ⬆

15. **What are different authentication methods?**

Use **API keys** if you expect developers to build internal applications that don't need to access more than a single user's data.
Use **OAuth** access tokens if you want users to easily provide authorization to applications without needing to share private data or dig through developer documentation.
Use **session cookies**, here server is responsible for creating a session for the particular user when the user log's in, after that the id of the session is stored in a cookie on the user browser. For every request sent by the user, the cookie will be sent too, where the server can compare the session id from the cookie with the session information stored on the server so the user identity is verified.

Back to Top ⬆

16. **What are disadvantages of using session based authentication?**

Compromised Secret Key : The best and the worst thing about JWT is that it relies on just one Key. Consider that the Key is leaked by a careless or a rogue developer/administrator, the whole system is compromised!
Cannot manage client from the server
Cannot push Messages to clients
Crypto-algo can be deprecated
Data Overhead : The size of the JWT token will be more than that of a normal Session token
Complicated to understand: JWT uses cryptographic Signature algorithms to verify the data and get the user-id from the token. Understanding the Signing Algo in itself requires basics of cryptography.

Back to Top ⬆

17. What are disadvantages of using jwt based authentication?

Session based authentication:
Because the sessions are stored in the server's memory, scaling becomes an issue when there is a huge number of users using the system at once.
Cookies normally work on a single domain or subdomains and they are normally disabled by browser if they work cross-domain (3rd party cookies). It poses issues when APIs are served from a different domain to mobile and web devices.

Back to Top ⬆

Table of Contents - Golang

No.	Questions
27	Why are goroutines light-weight?
28	If capacity is not defined in slice, what would the capacity be?
29	What is the easiest way to check if a slice is empty?
30	What is an advantage of Go evaluating implicit types at compile time?

1. What is Golang?

Golang is a statically typed, compiled programming language designed at Google by Robert Griesemer, Rob Pike, and Ken Thompson. Golang was designed at Google in 2007 to improve programming productivity in an era of multicore, networked machines and large codebases.

Back to Top ⬆

2. What are the pros and cons of Golang?

Pros:

- Ease of use
- A Smart Standard library
- Strong security built-in
- Garbage collected language.
- Minimalism & Readability
- Concurrency

Cons:

- No generics
- Error-handling boilerplate & lack of compile-time checks unhandled errors
- Shortage of high-level parallelism and concurrency features

Back to Top ⬆

3. What kind of projects are suitable to be built in Golang?

- Cloud services
- Media platforms
- Broadcast providers
- Projects with microservice architecture

Back to Top ⬆

4. Is Golang an object oriented language?

Golang has types and methods and allows an object-oriented style of programming, there is no type hierarchy.Golang has some properties of object oriented programming like Encapsulation , Composition , but it doesn't have inheritance , classes , function overloading .

Back to Top ⬆

5. What are the data types in Golang?

i. **Basic type**: Numbers, strings, and booleans .
ii. **Aggregate type**: Array and structs .
iii. **Reference type**: Pointers, slices, maps, functions, and channels .
iv. **Interface type**

Back to Top ⬆

6. Can you return multiple values from a function?

Yes. A Go function can return multiple values, each separated by commas in the return statement.

```go
package main

import "fmt"

func foo() (string, string) {
   return "foo", "ball"
}

func main() {
   fmt.Println(foo())
}
```

Back to Top ⬆

7. ### What is a GOPATH?

   The GOPATH environment variable specifies the location of your workspace. It defaults to a directory named go inside your home directory

   The command go env GOPATH prints the effective current GOPATH; it prints the default location if the environment variable is unset.

Back to Top ⬆

8. ### What are Goroutines?

   Goroutines are incredibly lightweight “threads” managed by the go runtime. They enable us to create asynchronous parallel programs that can execute some tasks far quicker than if they were written in a sequential manner.

Back to Top ⬆

9. ### What is nil in Go?

   nil is a predeclared identifier in Go that represents zero values for pointers, interfaces, channels, maps, slices and function types.

Back to Top ⬆

10. ### What is the difference between array and slice in Go ?

    - **ARRAY** : An array is a fixed collection of data. The emphasis here is on fixed, because once you set the length of an array, it cannot be changed.

```go
arr := [4]int{3, 2, 5, 4}
```

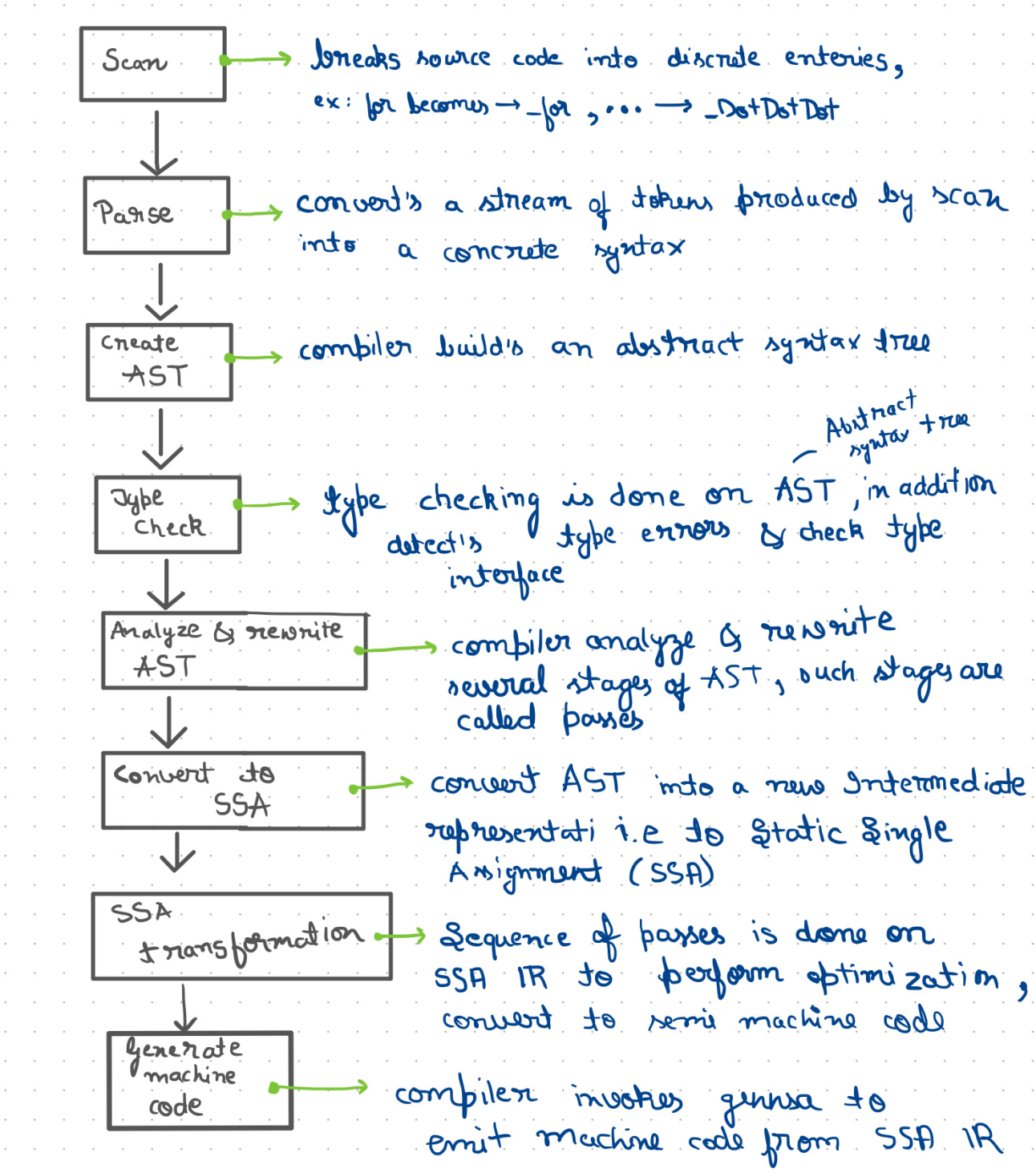
    - **SLICE** : Slices are much more flexible, powerful, and convenient than arrays. Unlike arrays, slices can be resized using the built-in append function .

```go
slicee := make([]Type, length, capacity)
```

Back to Top ⬆

11. ### How does a go compiler work?

    A Go compiler goes through the following steps , they are in brief , if we go in detail then you will need a complete book to understand each module, for interview purpose , I have attached a hand written note , I will generate a digital form soon



Back to Top ⬆

12. **What is an Interface and Why do you use it?**

The interface is a collection of methods as well as it is a custom type.

```
package main

    import "fmt"

    // Creating an interface
    type tank interface {

        // Methods
        Tarea() float64
        Volume() float64
    }

    type myvalue struct {
```

```
    radius float64
    height float64
}

// Implementing methods of
// the tank interface
func (m myvalue) Tarea() float64 {

    return 2*m.radius*m.height +
        2*3.14*m.radius*m.radius
}

func (m myvalue) Volume() float64 {

    return 3.14 * m.radius * m.radius * m.height
}

// Main Method
func main() {

    // Accessing elements of
    // the tank interface
    var t tank
    t = myvalue{10, 14}
    fmt.Println("Area of tank :", t.Tarea())
    fmt.Println("Volume of tank:", t.Volume())
}
```

Interfaces can make code clearer, shorter, more readable, and they can provide a good API between packages, or clients (users) and servers (providers).

Back to Top ⬆

13. What are concurrency and parralism and what is the difference between both?

```
**Concurrency** :
Defination 1 : Dealing with multiple things at once. <br/>
Defination 2 : A Composition of independently executing processes(Example: suppose there are two tasks A and B ,
the way this work is A task done 70% meanwhile it has to wait for something , so it picks up task B and try to
complete if suppose B task has to wait at 60% , for something then it picks up A task them completes it and comes
back to B )

**Parralism** :
 Defination 1 : Parallelism is about doing lots of things at once. <br/>
 Defination 2 : It is the simultaneous execution of (possibly related) computations. (Example: suppose there are
two tasks A and B , it takes both tasks and try to complete both together )

![cocurrency_parllel](/img/cocurrency_parllel.jpg)
```

Back to Top ⬆

14. What are the difference between goroutines and threads?

Threads : A thread is just a sequence of instructions that can be executed independently by a processor. Threads use a lot of memory due to their large stack and requires call to OS for resources (such as memory) which is slow. so doesn't always guarantee a better performance than processes in this multi-core processor world.

Goroutines:Goroutines exists only in the virtual space of go runtime and not in the OS. and A goroutine is created with initial only 2KB of stack size. Each function in go already has a check if more stack is needed or not and the stack can be copied to another region in memory with twice the original size. This makes goroutine very light on resources.

Back to Top ⬆

15. What are channels for?

Channels are the pipes that connect concurrent goroutines. You can send values into channels from one goroutine and receive those values into another goroutine.

Back to Top ⬆

16. Can you do something in goroutines using channels?

- Channels are goroutine-safe and can store and pass values between goroutines

- Channels provide FIFO semantics.
- Channels cause goroutines to block and unblock, which we just learned about.

Back to Top ⬆

17. ### What is a Closure?

A closure is a function value that references variables from outside its body. The function may access and assign to the referenced variables.

Back to Top ⬆

18. ### What are runtime and runtime packages?

The runtime library implements garbage collection, concurrency, stack management, and other critical features of the Go language. The Package runtime contains operations that interact with Go's runtime system, such as functions to control goroutines.

Back to Top ⬆

19. ### How can you get how many cores your computer has?

With the help of runtime package

```
package main

import (
     "fmt"
     "runtime"
       )

   func main() {
   fmt.Println(runtime.NumCPU())
   }
```

Back to Top ⬆

20. ### How would you tell a goroutine to use less core than what you have?

We can restrict the number of goroutines running at the same time , like below

```
 package main

 import (
 "flag"
 "fmt"
 "time"
 "sync"
)

// Fake a long and difficult work.
func DoWork() {
  time.Sleep(500 * time.Millisecond)
}

func main() {
  maxNbConcurrentGoroutines := flag.Int("maxNbConcurrentGoroutines", 2, "the number of goroutines that are allc
  nbJobs := flag.Int("nbJobs", 5, "the number of jobs that we need to do")
  flag.Parse()

  concurrentGoroutines := make(chan struct{}, *maxNbConcurrentGoroutines)

  var wg sync.WaitGroup

  for i := 0; i < *nbJobs; i++ {
      wg.Add(1)
      go func(i int) {
          defer wg.Done()
          concurrentGoroutines <- struct{}{}
          fmt.Println("doing", i)
          DoWork()
          fmt.Println("finished", i)
          <-concurrentGoroutines
      }(i)
  }
```

```
    wg.Wait()
}
```

Back to Top ⬆

21. ## How would you determine the type of a variable and Which package to use for it?

    There are three different ways to find type of variable in Golang

    - **reflect.TypeOf Function**: Using the golang inbuilt package reflect we can find the Type of variable

```
package main


import (
    "fmt"
    "reflect"
)

func main(){
var string_type =  "Hello Go";
var complex_type =  complex(9, 15);

 fmt.Println("string_type", reflect.TypeOf(string_type))
 fmt.Println("complex_type = ", reflect.TypeOf(complex_type))
}
```

    - **reflect.ValueOf.Kind() Function** : Using the golang inbuilt package reflect we can find the Type of variable

```
package main


import (
    "fmt"
    "reflect"
)

func main(){
var string_type =  "Hello Go";
var complex_type =  complex(9, 15);

 fmt.Println("string_type", reflect.ValueOf(string_type).Kind())
 fmt.Println("complex_type = ", reflect.TypeOf(complex_type).Kind())
}
```

    - **%T with Printf** : You can use Printf also to find value of variable

```
package main


import (
    "fmt"
    "reflect"
)

func main(){
var string_type =  "Hello Go";
var complex_type =  complex(9, 15);

 fmt.Printf("string_type=%T\n", string_type)
 fmt.Printf("complex_type =%T\n", complex_type)
}
```

    Back to Top ⬆

22. ## What all types can map store?

    Value - The Value type of a mpa can be anything, including another map
    Key - The key type of Map can only be values that can be compared i.e - boolean, numeric, string, pointer, channel, and interface types, and structs or arrays , that excludes - slices, maps, and functions

    Back to Top ⬆

23. ### What are microservices?

Microservices is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

Back to Top ⬆

24. ### Why are there no classes in Go ?

Go doesn't require an explicit class definition as Java, C++, C#, etc do. Instead, a "class" is implicitly defined by providing a set of "methods" which operate on a common type. The type may be a struct or any other user-defined type. For example:

```
type Integer int;
  func (i *Integer) String() string {
      return strconv.itoa(i)
  }
```

is analogous to below code in java:

```
class Integer {
      public int i;
      public String toString() { return Integer.toString(i); }
  }
```

Back to Top ⬆

25. ### Difference between Compile time and runtime?

Compile-time is the time at which the source code is converted into an executable code
Run time is the time at which the executable code is started running.

Back to Top ⬆

26. ### How to generate a true random number in golang?

The default number generator is deterministic, so it'll produce the same sequence of numbers each time by default , so you need to seed it with different number you can do it with nano seconds like below

```
package main
import (
   "fmt"
   "math/rand"
    "time"
   )
func main(){
s1 := rand.NewSource(time.Now().UnixNano())
r1 := rand.New(s1)
fmt.Print(r1.Intn(100))
}
```

Back to Top ⬆

27. ### Why are goroutines light-weight?

A goroutine is created with initial only 2KB of stack size. Each function in go already has a check if more stack is needed or not and the stack can be copied to another region in memory with twice the original size. This makes goroutine very light

Back to Top ⬆

28. ### If capacity is not defined in slice, what would the capacity be?

It would be set by default to length of the slice

Back to Top ⬆

29. **What is the easiest way to check if a slice is empty?**

You can use the length (len(slice)) property to find if the slice is empty

```
if(len(slice_data)==0){
fmt.Println("Empty slice")
}
```

Back to Top ⬆

30. **What is an advantage of Go evaluating implicit types at compile time?**

Types can be implicitly inferred from expressions and don't need to be explicitly specified.The special handling of interfaces and the implicit typing makes Go feel very lightweight and dynamic.

Back to Top ⬆

Table of Contents - Database Engineering

1. **What is ACID Model in Database?**

The ACID database transaction model ensures that a performed transaction is always consistent. This makes it a good fit for businesses which deal with online transaction processing (e.g: finance institutions) or online analytical processing (e.g: data warehousing). These organizations need database systems which can handle many small simultaneous transactions. There must be zero tolerance for invalid states

Back to Top ⬆

2. **What does ACID acronym mean?**

- A - Atomicity
- C - Consistency
- I - Isolation
- D - Durability

3. **Explain in brief the ACID Model?**

- **Atomicity**: Each transaction is either properly carried out or the process halts and the database reverts back to the state before the transaction started. This ensures that all data in the database is valid .
- **Consistency**: The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database , also a DB should have Atomicity and Isolation for this property.
- **Isolation**: Any reads or writes performed on the database should not be impacted by other reads and writes of separate transactions occurring on the same database.
- **Durability**: The database should be durable enough to hold all its latest updates even if the system fails or restarts .

Back to Top ⬆

4. **What are the read phenomena in Isolation?**

Releases

No releases published

Packages

No packages published

Languages

- **JavaScript** 100.0%