Google Developer Student Clubs

Code With Node 2.0

GDSC SRM

Gautam J
@gautam.j

About Myself

Gautam J

- 3rd Year in B.Tech Artificial Intelligence
- Technical Lead at Google Developer Student Clubs SRM
- Working as an AI Intern for the past 6 months
- 4+ years of experience in Machine Learning and Web Development
- I love Open Source and Linux
- Earned more than 570 stars and 130 forks on GitHub

Code With Node 2.0

Prerequisite

- Laptop with internet connection
- Node.js - Version 16 or above
- Visual Studio Code
- Postman
- Experience in any programming language (even C works!)

Code With Node 2.0

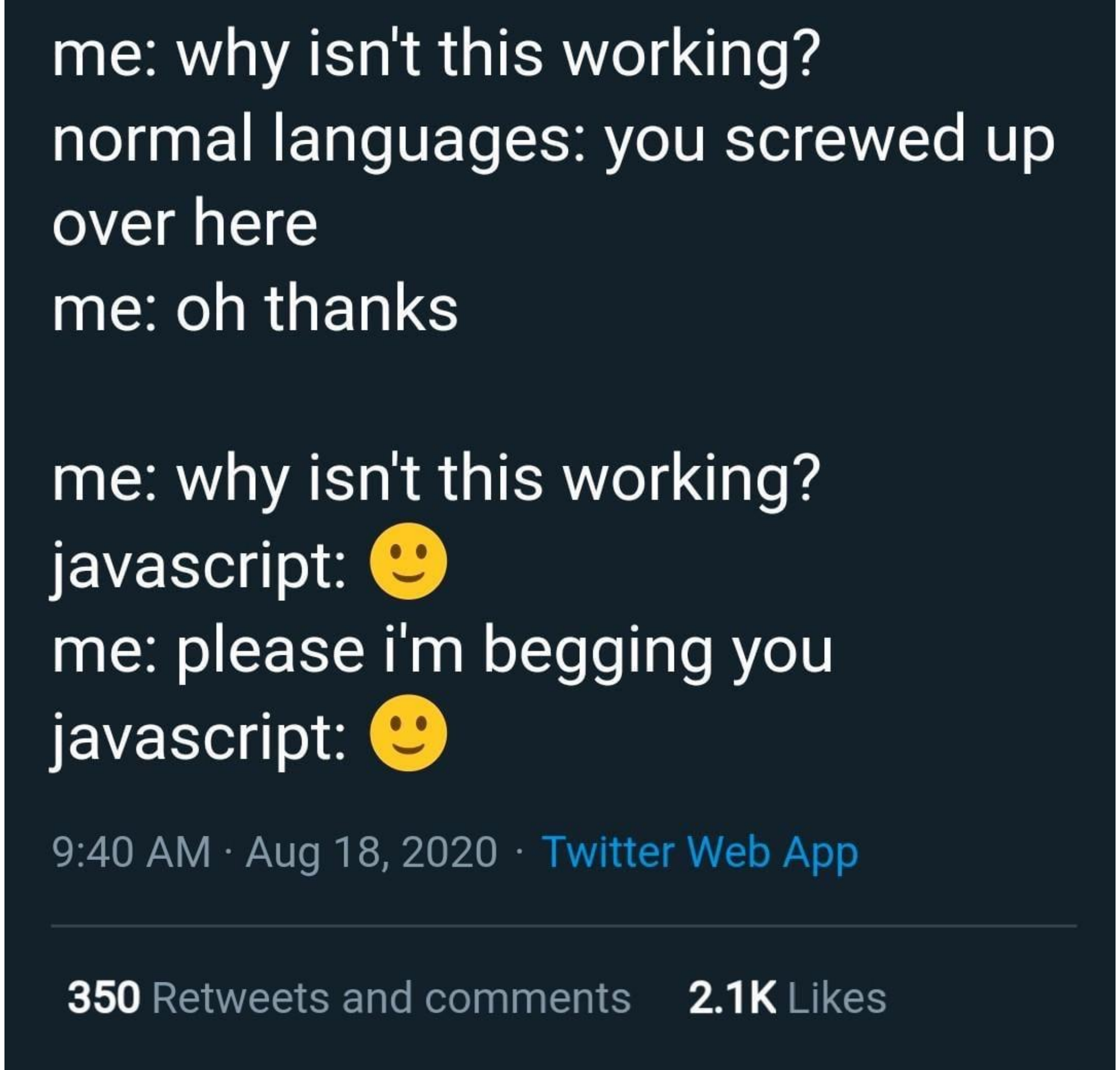
What you'll learn

- Basics of JavaScript
- Basics of Node.js
- Overview of how web works
- Basics of Database using FireStore
- Basics of Express.js
- CRUD Application
- Backend Developer

Basics of JavaScript

Why JavaScript?

It is easy to learn and understand. It has a simple syntax when compared to other programming languages. Almost all modern day browsers support JavaScript, making it perfect for Web Development.



me: why isn't this working?
normal languages: you screwed up over here
me: oh thanks

me: why isn't this working?
javascript: 🙂
me: please i'm begging you
javascript: 🙂

9:40 AM · Aug 18, 2020 · Twitter Web App

350 Retweets and comments **2.1K** Likes

The only way to learn a new programming language is by writing programs in it.

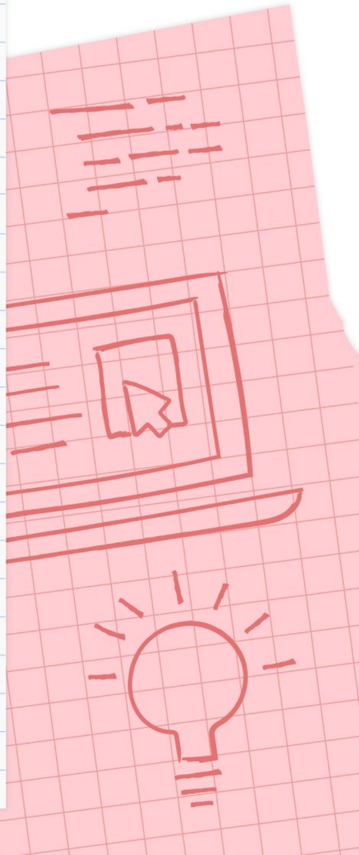
- Dennis Ritchie

```
filterStudies({ studies, filterByOrg = false, filte
studies.filter(study => {
organizat
```

Basics of JavaScript

Time to Code!

Let’s build our first “Hello World” application using JavaScript!



Me coding by myself

Me pair coding while sharing my screen

Basics of JavaScript

Let's code these!

- Print Statements, Comments, Declaring Variables, Scope
- Arithmetic, Relational, and Logical Operations, If-Else Statements
- For, While, and Do-While Loops, Switch Case Statements, Ternary Operators
- Template Literals, Arrow Functions, Arrays and Objects, Filter and Map

```
// Explain the code
let result = (“b" + “a” + + “a” + “a”).toLowerCase();
console.log(result);
```

When you add a string to an integer and wait for an error but javascript returns a string

Basics of Node.js

What is Node.js?

Node.js is an Open Source Cross-Platform JavaScript runtime environment. It is built on Google Chrome's V8 Engine. It is a single-threaded, non-blocking, and event-driven environment, and supports asynchronous programming. This allows it to be very fast and highly scalable.

Additionally, Node.js comes with the Node Package Manager (NPM). It can be used to download packages that have pre-written code, making development easier.

Basics of Node.js

Single-Threaded

Node.js follows a Single-Thread with Event Loop Model. The Event Loop is an infinite loop that runs continuously as long as your application is running. The Event Loop runs on a Single Thread. It can execute only one function at a time. It handles concurrent requests using Callbacks.

A multi-threaded application would create a thread for every new request, thereby making it resource intensive.

Basics of Node.js

Asynchronous Programming

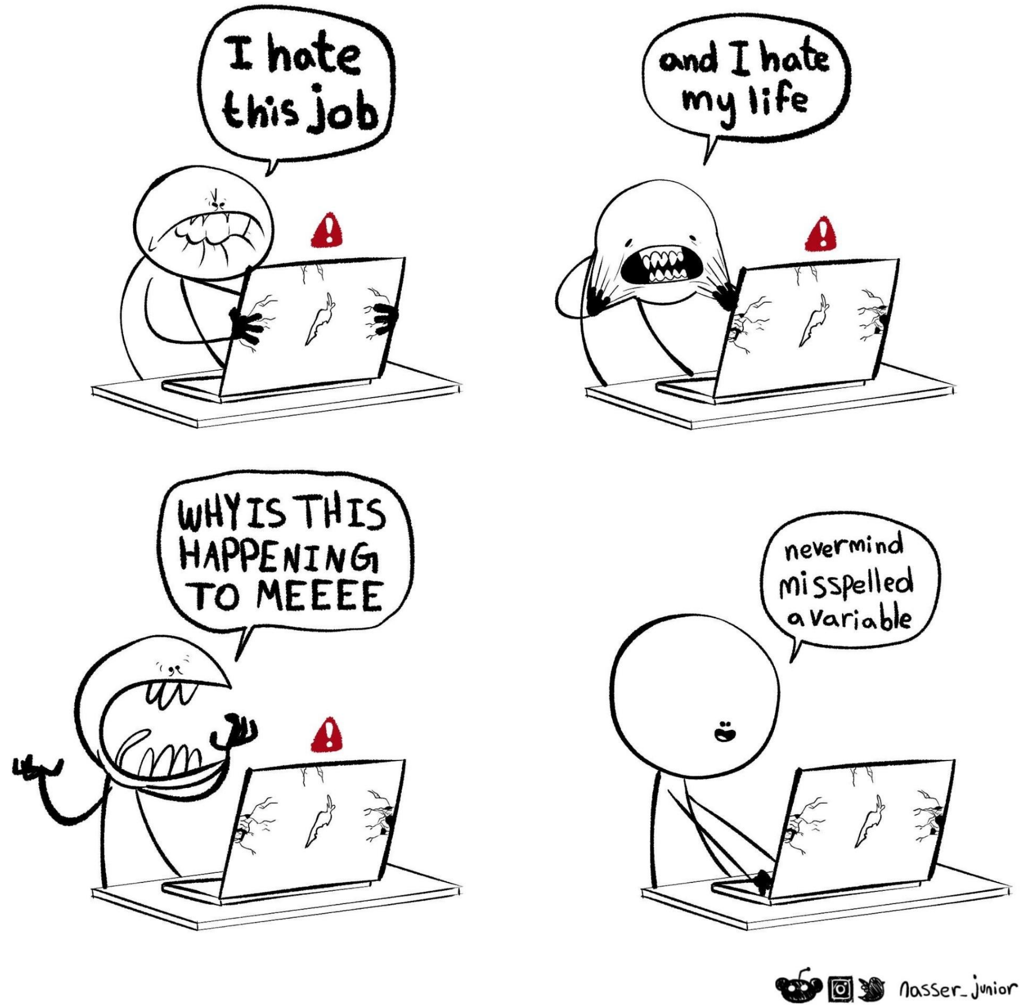
Asynchronous programming is a technique that enables your program to start a potentially long-running task and still be able to be responsive to other events while that task runs, rather than having to wait until that task has finished. Once that task has finished, your program is presented with the result.

In synchronous programming, the program has to wait for each line of code to get completed before executing the next line of code. Thus, if a single line of code takes a lot of time to execute (e.g., copy a large file), the entire program is frozen, and cannot execute any other lines of program.

Basics of Node.js

Non-Blocking Event Driven Model

Node.js pushes all the requests it receives into a “Queue”. It then pops the requests one at a time, and resolves it. The output is returned to the user either at the start or end of a loop in the Event Loop Cycle. This is done using Callbacks, and makes sure that the application is still usable even when resolving requests. This makes Node.js a Non-Blocking Event Driven Model.


I hate this job
and I hate my life
WHY IS THIS HAPPENING TO MEEEE
nevermind misspelled a variable
nasser_junior

Basics of Node.js

What is a Callback?

When a function is passed as an argument to another function, it is called a callback function. This callback function can be invoked or “called” by the outer function to which it was passed, whenever required.

This can happen in both synchronous and asynchronous manner.

Google Developer Student Clubs

Basics of Node.js

What is a Promise?

A Promise is a proxy for a value which is not necessarily known when the promise is created. Simply put, instead of immediately returning the value, it “promises” to supply the value at some point in the future.

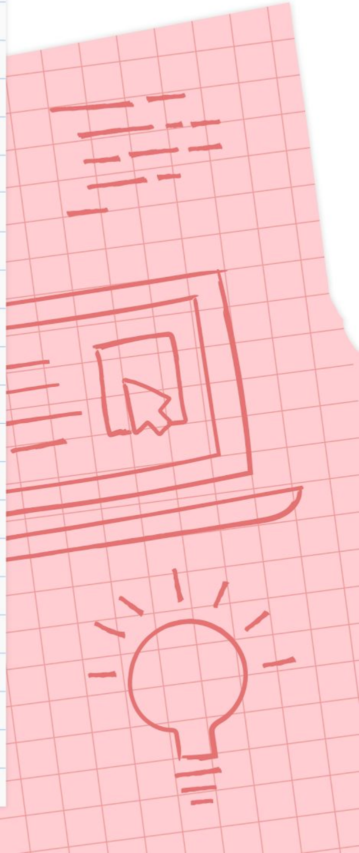
Promises are the ideal choice for handling asynchronous operations in the simplest manner. It is a much better technique than Callbacks.

Basics of Node.js

Async Await Technique

The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains.

Async operates asynchronously via the event loop. Async functions will always return a value. Await makes the code wait until the promise returns a result. It only makes the async block wait.

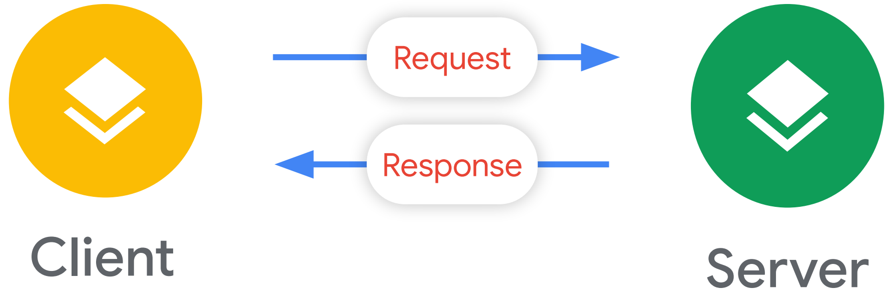


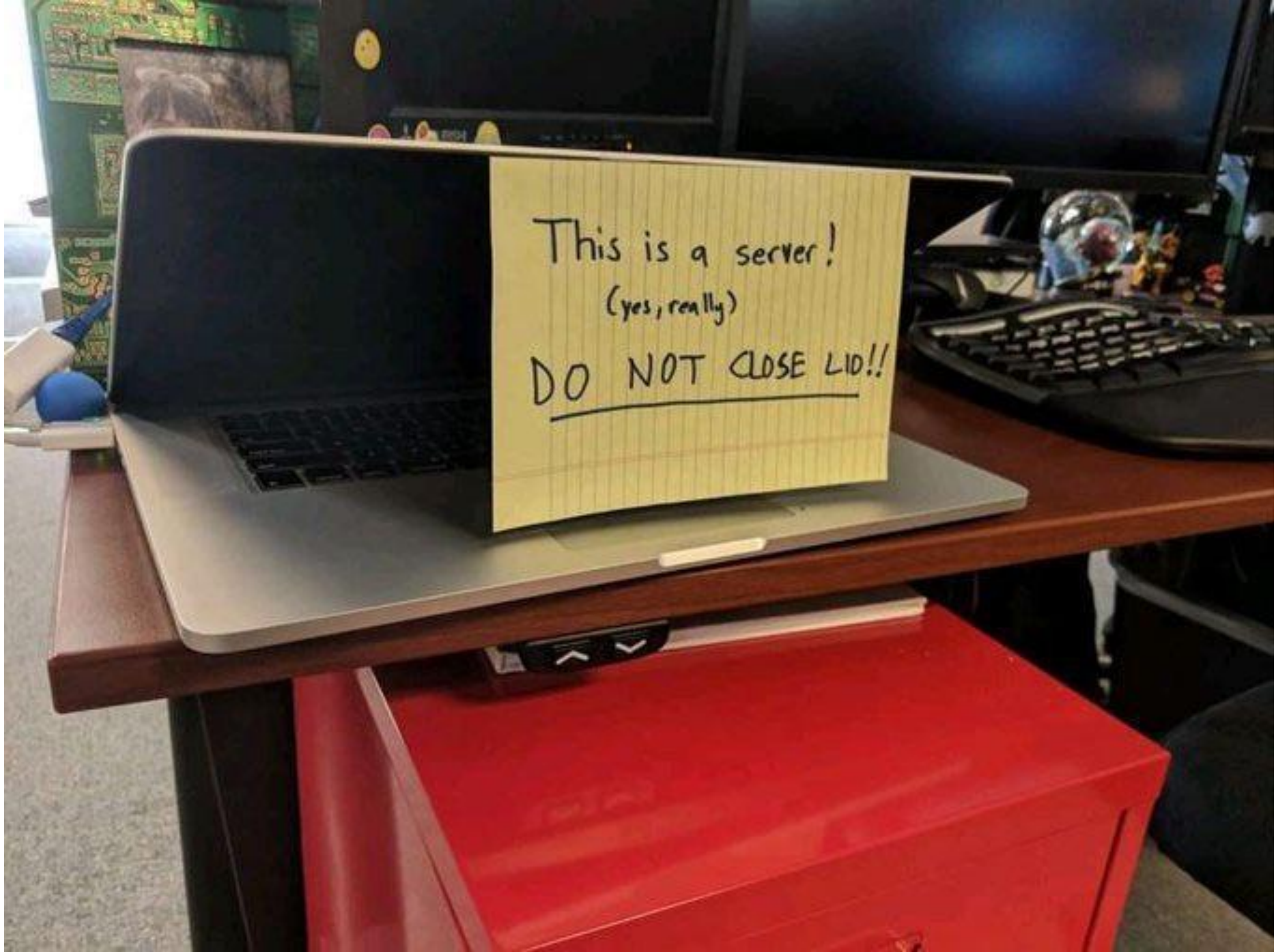
How Web Works?

Client and Server

Computers connected to the internet are called clients and servers.

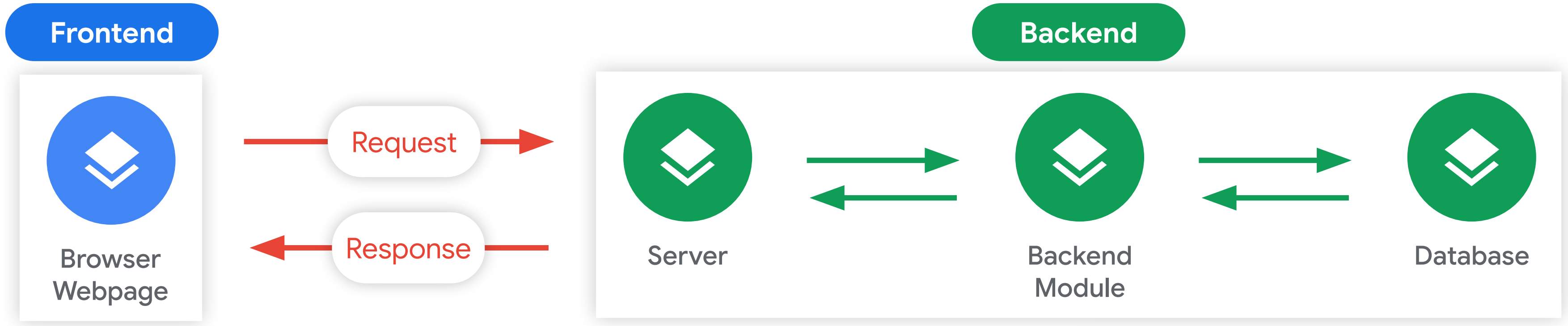
- Clients are the user's internet-connected devices, e.g. your phone, laptop browser.
- Servers are computers that store web pages, sites, or apps.
- When a client wants to access a webpage it sends a request. A copy of the webpage is downloaded from the server and sent to the client machine as response, which is displayed in the user's web browser.

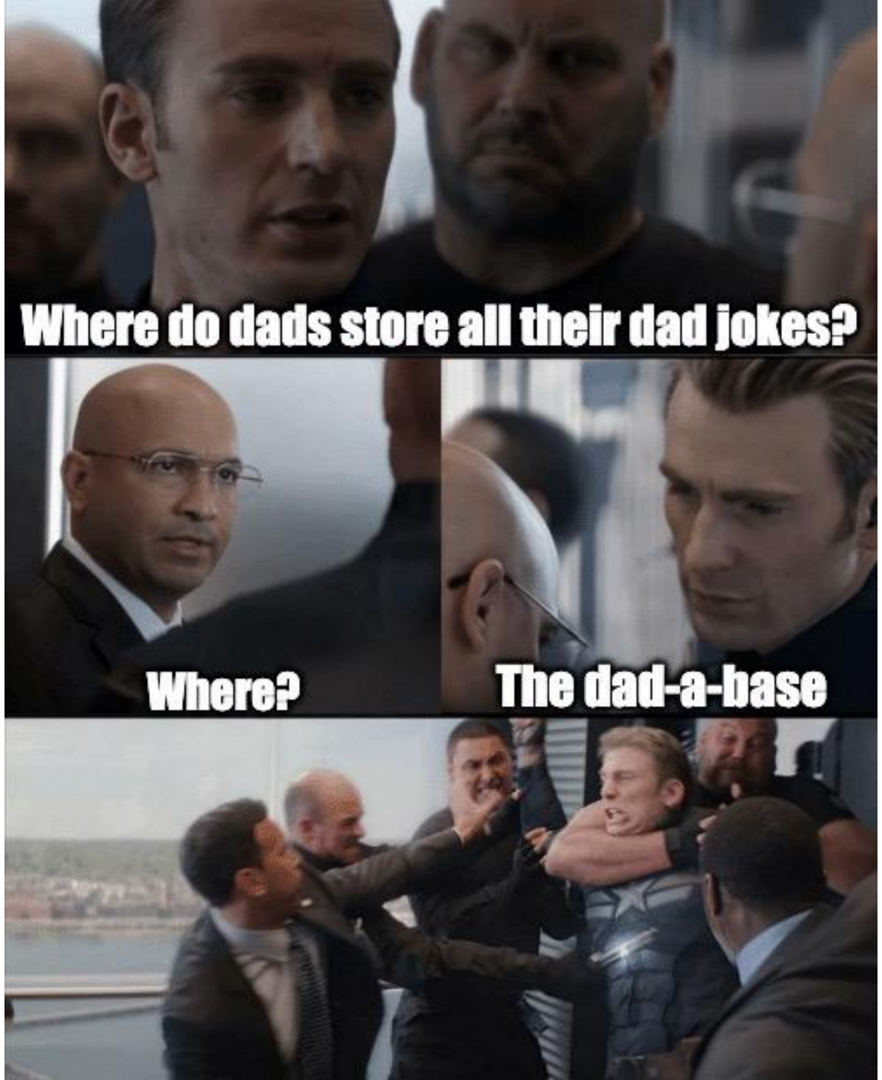



This is a server!
(yes, really)
DO NOT CLOSE LID!!

How Web Works?

Frontend and Backend




Where do dads store all their dad jokes?
Where?
The dad-a-base

Basics of Database

What is FireStore?

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Google Cloud. It is a NoSQL database that keeps data in sync across multiple clients in real time.

Basics of Database

What is NoSQL?

NoSQL databases are non-tabular databases that store data in formats other than the typical relational table. In Firebase, data is stored in documents, which themselves are organized into collections. This allows them to be very flexible.

Since NoSQL databases use sharding for horizontal scaling, they are highly efficient and scalable.

Basics of Database

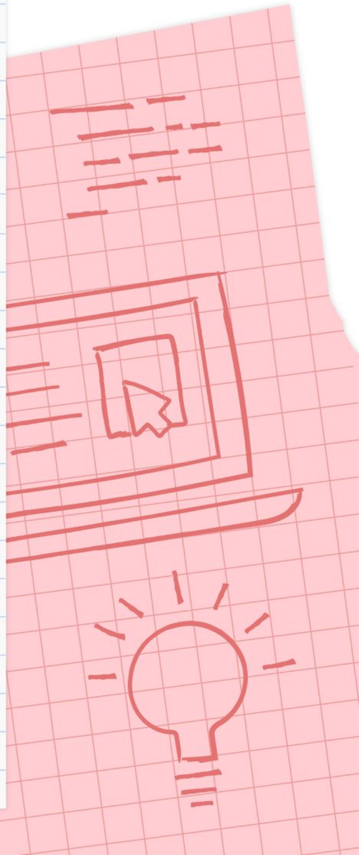
What is a Document?

A document is a record in a database. It stores data in key-value pairs, and is very similar to an “Object” in JavaScript. A document typically stores information about one object and any of its related metadata.

Basics of Database

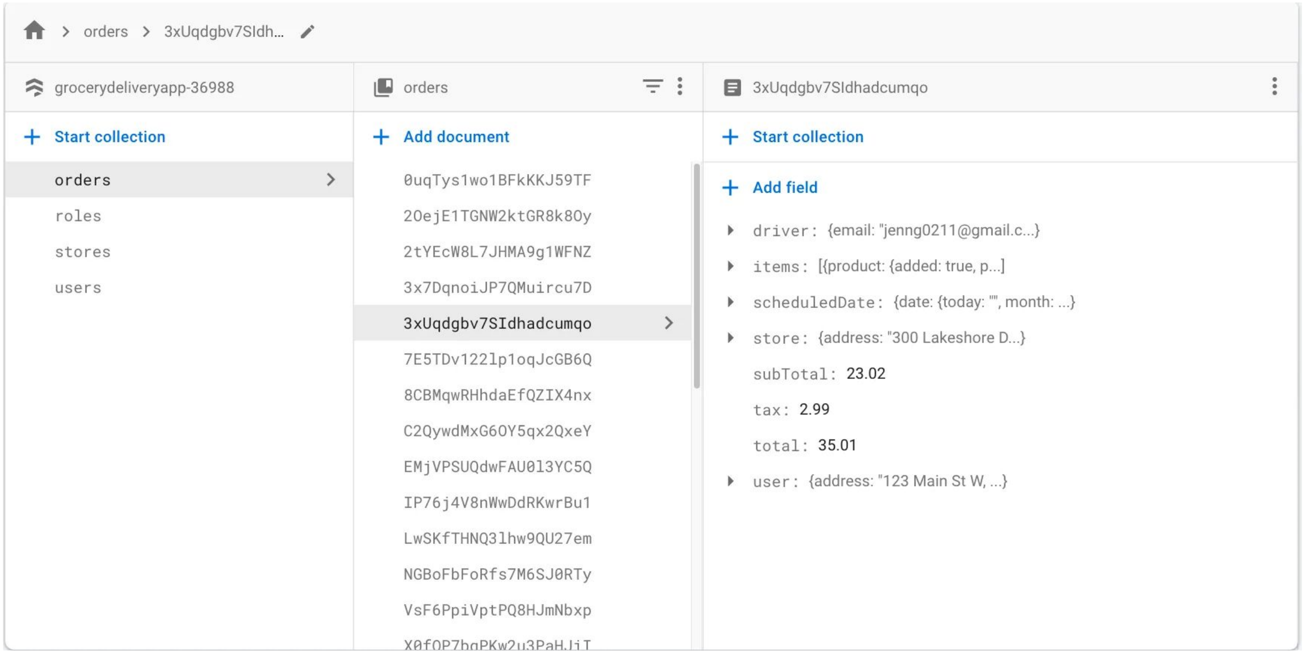
What is a Collection?

A collection is a group of documents. Collections typically store documents that have similar contents. Note that not all documents in a collection are required to have the same fields, because NoSQL databases have a flexible schema.



Basics of Database

Example of a Firestore Database



orders > 3xUqdgbv7SIdh...

grocerydeliveryapp-36988

+ Start collection

orders
roles
stores
users

orders

+ Add document

0uqTys1wo1BFkKJ59TF
2OejE1TGNW2ktGR8k8Oy
2tYEcW8L7JHMA9g1WFNZ
3x7DqnoiJP7QMuircu7D
3xUqdgbv7SIdhadcumqo
7E5TDv122lp1oqJcGB6Q
8CBMqwRHhdaEfQZIX4nx
C2QywdMxG6OY5qx2QxeY
EMjVPSUQdwFAU0l3YC5Q
IP76j4V8nWwDdRKwrBu1
LwSKfTHNQ3lhw9QU27em
NGBoFbFoRfs7M6SJ0RTy
VsF6PpiVptPQ8HJmNbxp
X0fOP7hqPKw2u3PaHJiT

3xUqdgbv7SIdhadcumqo

+ Start collection

+ Add field

```
driver: {email: "jenng0211@gmail.c...}
items: [{product: {added: true, p...]
scheduledDate: {date: {today: "", month: ...}
store: {address: "300 Lakeshore D...}
subTotal: 23.02
tax: 2.99
total: 35.01
user: {address: "123 Main St W, ...}
```

Google Developer Student Clubs

Basics of Database

What are CRUD Operations?

- Create: A new document is created inside a database. Each document has a unique identifier.
- Read: Documents already stored in the database can be read. Optionally, we can filter out what documents we need.
- Update: Existing documents in the database can be updated — either in whole or in part.
- Delete: Documents can be deleted from the database.

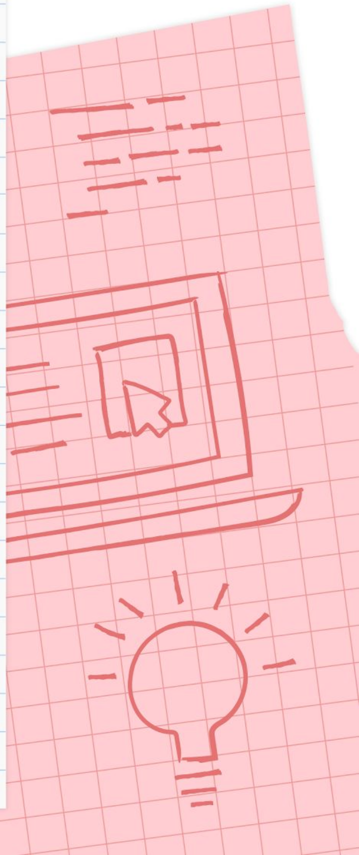
People liking snapchat because it doesn't save the messages
The developer that never learned SQL

Q&A Session

Shoot your questions!

You may visit the link shown (when enabled) to ask your questions. We will try our best to answer as many questions as possible within the time limit.

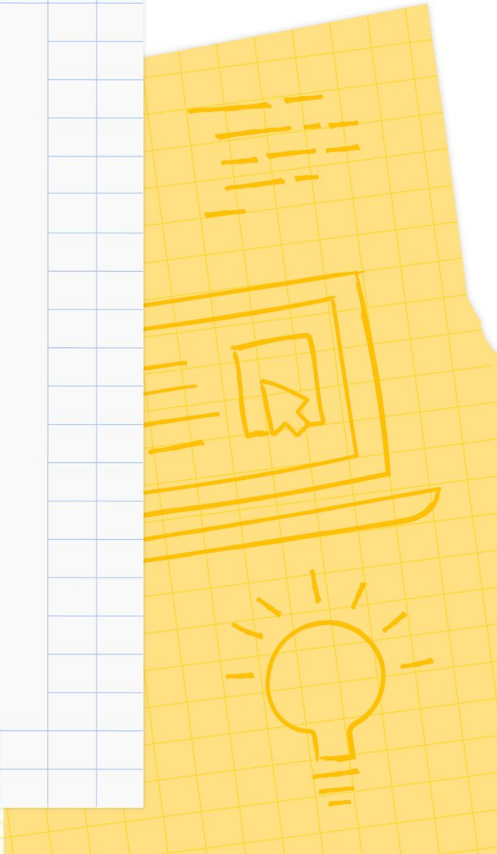
In case of programming errors you may post your code, along with the error message.



Quick Break

Refreshments are waiting!

Please take a quick break, we'll resume in a short while. Feel free to munch on the refreshments provided. Come back fresh for a major hands-on session, wherein you'll build an entire backend service.



Connect With Me

`linktr.ee/gautam_j`

You may visit the link to access my LinkedIn and Github profiles.

Additionally, you may also scan the QR code for the same.

Gautam J
@gautam.j

Basics of Express.js

What is Express.js?

Express is a minimal Node.js framework that provides a set of features for web applications. It allows us to write RESTful APIs in a quick and easy manner. This allows our backend service to communicate with the database effectively.

More specifically, it provides the functionality of routing and middleware. Since it also supports template rendering, we can use Express.js to create a frontend for our application as well.

Google Developer Student Clubs

Basics of Express.js

What is an API?

Application Programming Interface (API) enables two different software components to communicate with each other using a set of definitions and protocols. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses.

APIs simplify how developers integrate new application components into an existing architecture.

Basics of Express.js

What is a REST API?

REST stands for Representational State Transfer. REST APIs communicate via HTTP requests to perform standard database functions like CRUD. They provide a flexible, lightweight way to integrate applications.

The main feature of REST API is statelessness. Statelessness means that servers do not save client data between requests.

Basics of Express.js

HTTP Requests

- GET: Used to read or “get” data from a web server.
- POST: Used to send data (file, form data, etc.) to the server.
- DELETE: Used to delete a particular data on the server.
- PUT: Used to modify existing data on the server. It replaces the entire content of the data.
- PATCH: Similar to PUT, but it modifies only a part of the existing data.

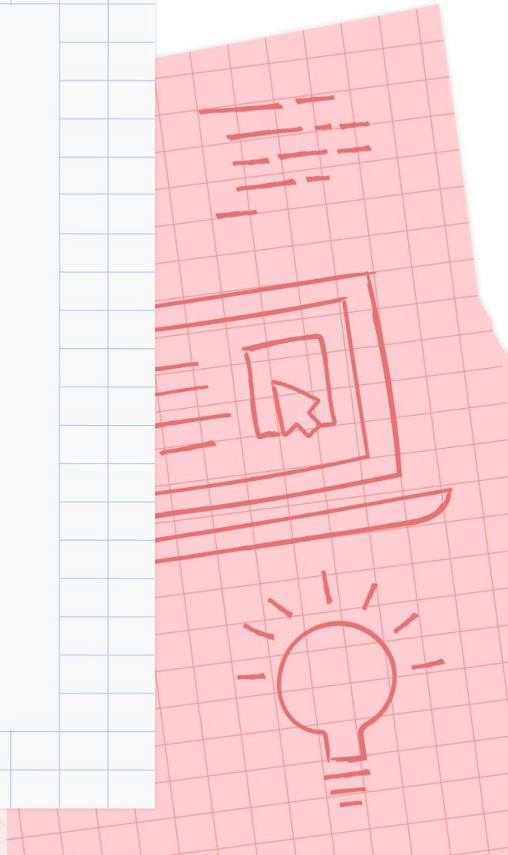
ChatGPT
every dev right now
Stack Overflow
Documentations
made with mematic

Google Developer Student Clubs

Basics of Express.js

Time to Code!

Let’s build our first “Hello World” application using Express.js!



Basics of Express.js

What is Routing?

Routing is how an API matches a URI to an action. It is the mechanism by which requests (as specified by a URI and HTTP method) are routed to the code that handles them. A router handles all the routing in an application. Simply put, a router determines what should happen when a user visits a certain page, or performs a certain action on the webpage.

In technical terms, it connects the logic to a specific URI and HTTP request.

Basics of Express.js

What is a Middleware?

Middleware is software that different applications use to communicate with each other. Without middleware, we would have to build APIs for each software component that connects to the application.

Simply put, a middleware is a piece of code that runs between each request made by the client. It sits in the “middle” of the communication, and hence the name middleware.

Basics of Express.js

What is MVC Architecture?

The Model-View-Controller (MVC) framework is an design pattern that separates an application into three main logical components — Model, View, and Controller.

The View component handles all the UI logic of the application.

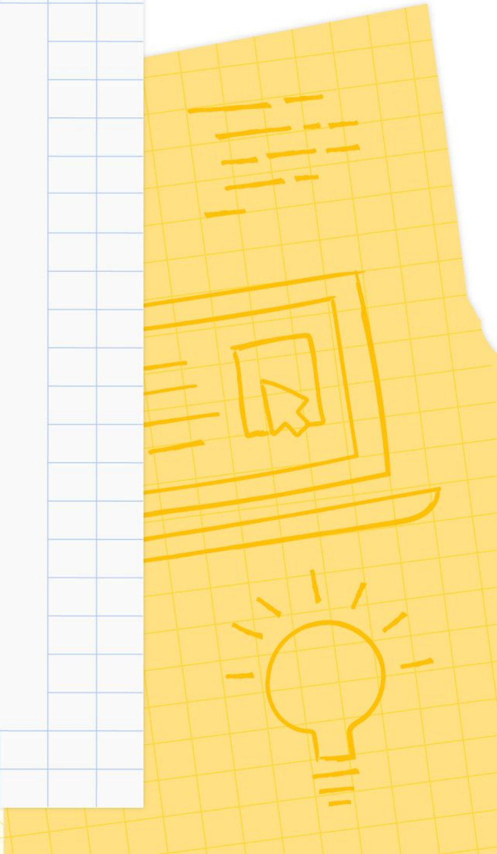
The Model component handles all the data-related logic of the application.

The Controller component enables the interconnection between the View and the Model component.

CRUD Application

Let's Code!

We shall build a fully functional backend service for a Blog Application, that supports CRUD operations using Node.js, Express.js, and FireStore.



Programmers when they finally get a different error message

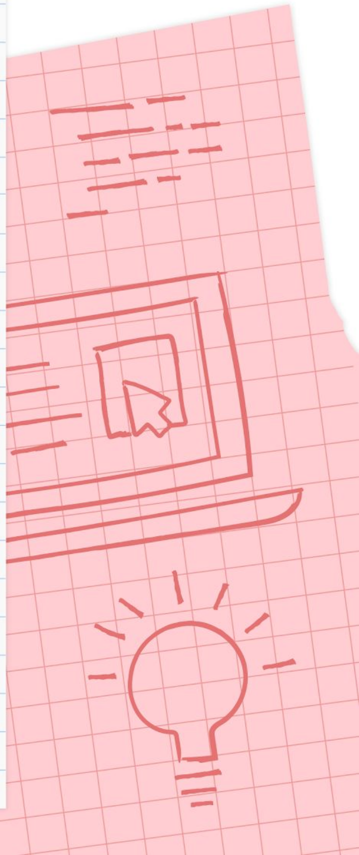
Developers at the beginning of a project. vs. Developers at the end of a project.

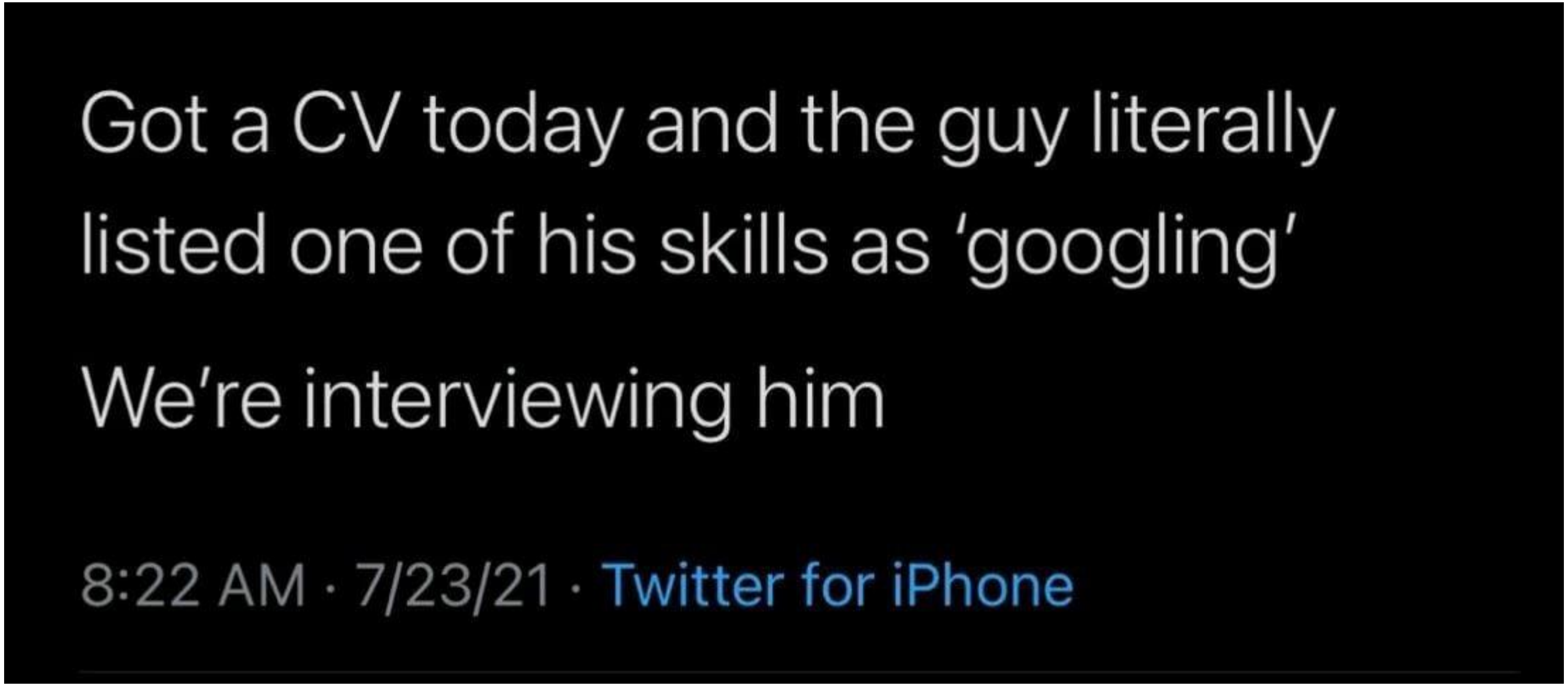
Q&A Session

Shoot your questions!

You may visit the link shown (when enabled) to ask your questions. We will try our best to answer as many questions as possible within the time limit.

In case of programming errors you may post your code, along with the error message.





Got a CV today and the guy literally listed one of his skills as 'googling'

We're interviewing him

8:22 AM · 7/23/21 · Twitter for iPhone

Thank You for Joining Us Today

`linktr.ee/gautam_j`

You may visit the link to access my LinkedIn and Github profiles.

Additionally, you may also scan the QR code for the same.

Gautam J
@gautam.j

Google Developer Student Clubs