

COMPLETE



Introduction to NodeJS



CERTIFICATE

NOTES

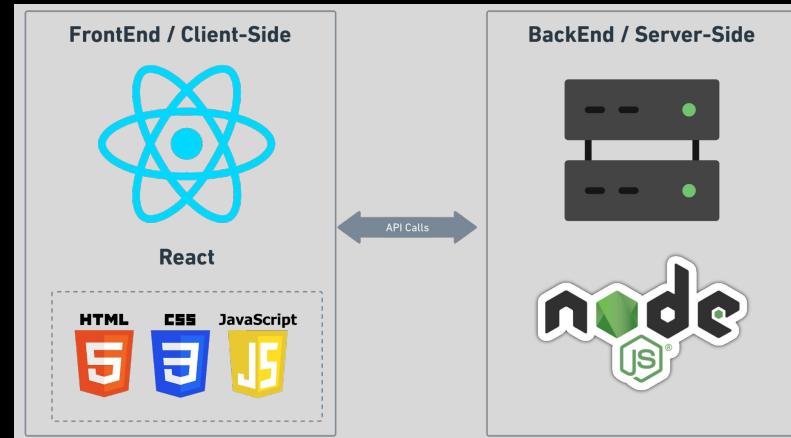
Ex-amazon Microsoft

You Tube [Playlist Link](#)



NodeJS

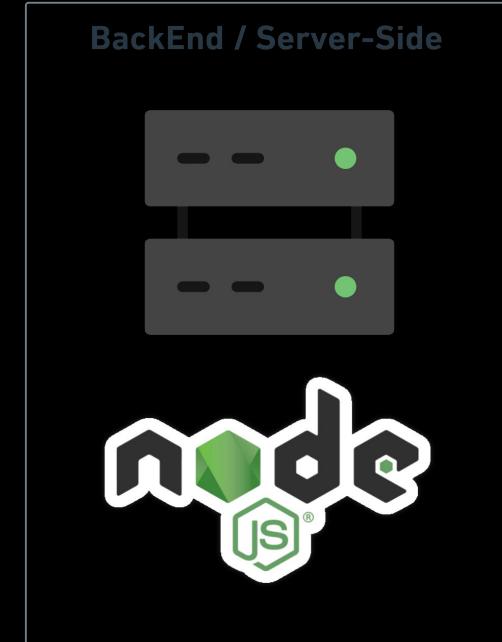
Complete Course





Introduction to NodeJS

1. Pre-requisites
2. What is NodeJS
3. NodeJs Features
4. JavaScript on Client
5. JavaScript on Server
6. Client Code vs Server Code
7. Other uses of NodeJs
8. Server architecture with NodeJs





JS is required for NodeJS

COMPLETE JS JAVASCRIPT

14 HOURS



MYNTRA
PROJECT



CERTIFICATE

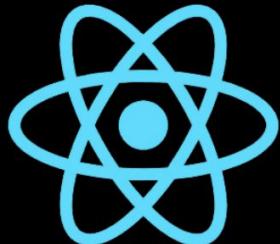
NOTES





React is recommended before NodeJS

COMPLETE



React



REDUX

20 HOURS

6 PROJECTS

B using
Bootstrap

CERTIFICATE

NOTES





2.What is NodeJS

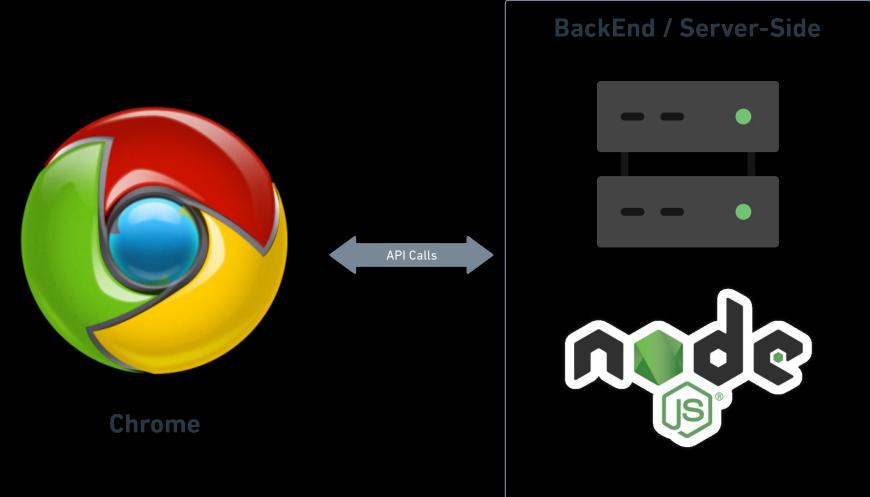


1. **JavaScript Runtime:** Node.js is an open-source, cross-platform runtime environment for executing JavaScript code outside of a browser.
2. **NodeJs** is a JavaScript in a different environment means Running JS on the server or any computer.
3. **Built on Chrome's V8 Engine:** It runs on the V8 engine, which compiles JavaScript directly to native machine code, enhancing performance.
4. V8 is written in C++ for speed.
5. V8 + Backend Features = NodeJs



2.What is NodeJS

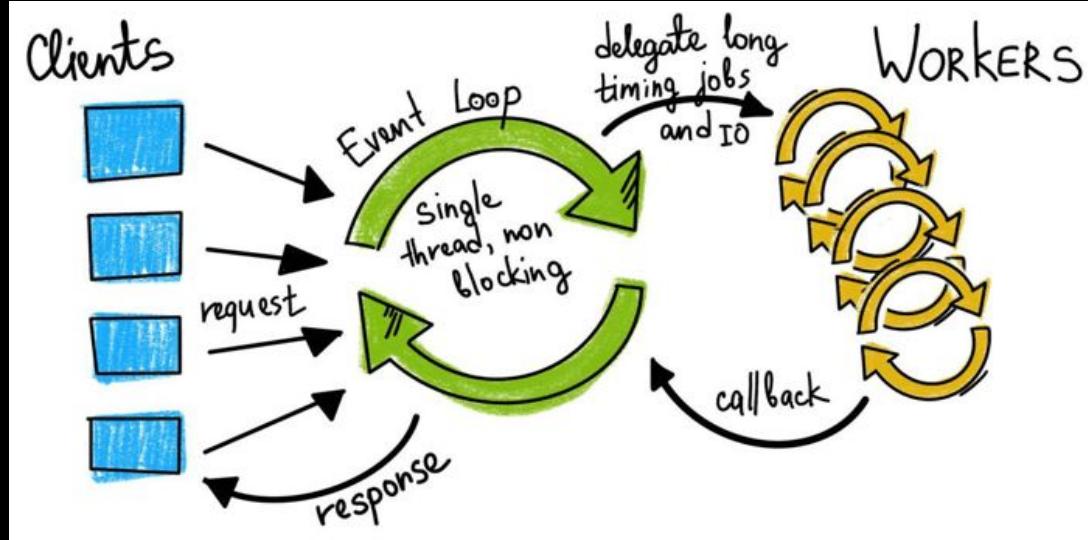
- Design:** Features an **event-driven**, **non-blocking I/O** model for efficiency.
- Full-Stack JavaScript:** Allows using JavaScript on both **server** and **client** sides.
- Scalability:** Ideal for **scalable** network applications due to its architecture.
- Versatility:** Suitable for web, real-time chat, and **REST API servers**.





3. NodeJS Features

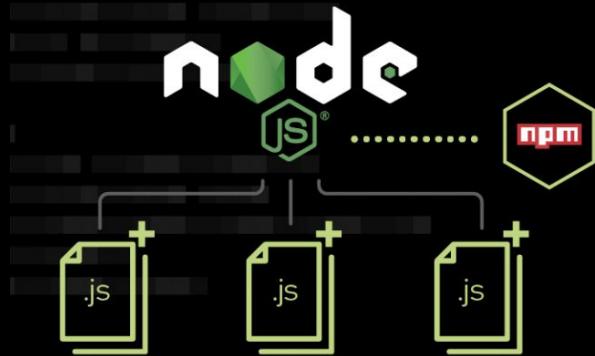
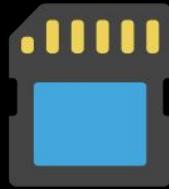
(Added)



1. **Non-blocking I/O:** Designed to perform non-blocking operations by default, making it suitable for I/O-heavy operations.
2. **Networking Support:** Supports TCP/UDP sockets, which are crucial for building lower-level network applications that browsers can't handle.



3. NodeJS Features (Added)

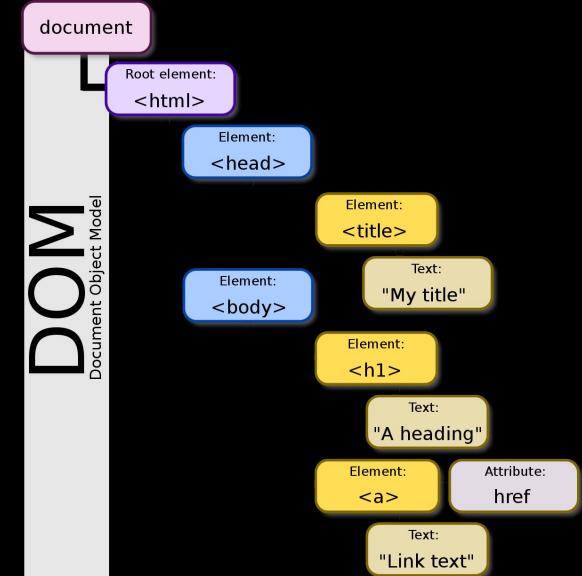
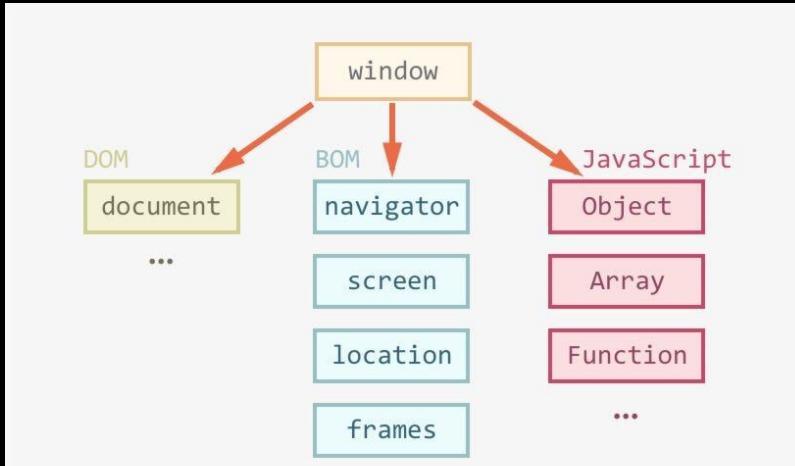


1. **File System Access:** Provides APIs to **read** and **write files** directly, which is **not possible in browser environments** for security reasons.
2. **Server-Side Capabilities:** Node.js enables JavaScript to run on the server, **handling HTTP requests**, **file operations**, and other server-side functionalities.
3. **Modules:** Organize code into **reusable modules** using `require()`.



3. NodeJS Features

(Removed)

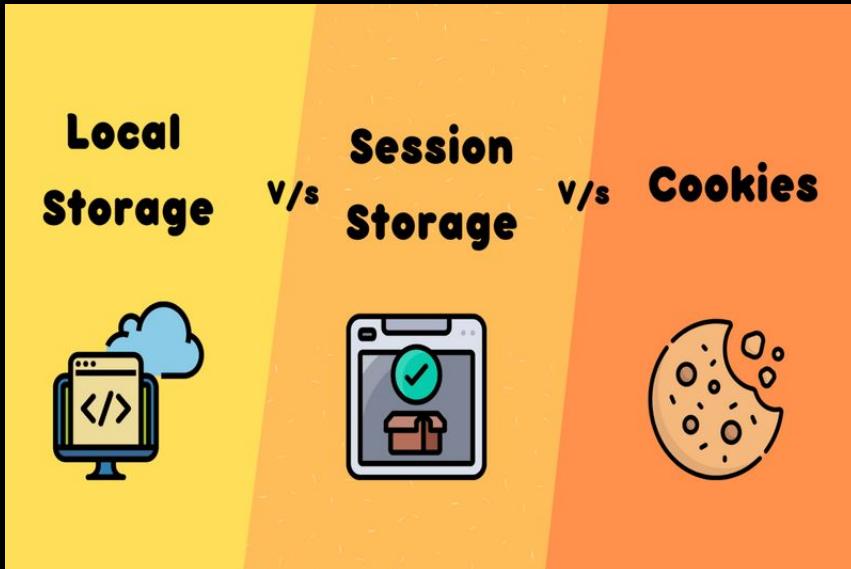


- 1. Window Object:** The global **window object**, which is part of web browsers, is absent in **Node.js**.
- 2. DOM Manipulation:** **Node.js** does not have a built-in Document Object Model (DOM), as it is not intended to interact with a webpage's content.
- 3. BOM (Browser Object Model):** No direct interaction with things like **navigator** or **screen** which are part of **BOM** in browsers.



3. NodeJS Features

(Removed)



A screenshot of the Chrome DevTools Application tab. The sidebar on the left lists "Manifest", "Service workers", and "Storage". The main area shows "App Manifest (unknown)" under "Application", and "Local storage", "Session storage", "IndexedDB", "Cookies", "Private state tokens", "Interest groups", "Shared storage", and "Cache storage" under "Storage".

Web-Specific APIs: APIs like `localStorage`, `sessionStorage`, and `fetch` are not available in Node.js.



4. JavaScript on Client

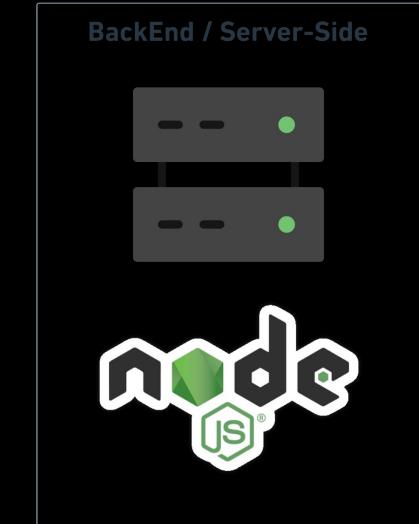


1. **Displays Web Page:** Turns HTML code into what you see on screen.
2. **User Clicks:** Helps you interact with the web page.
3. **Updates Content:** Allows changes to the page using JavaScript.
4. **Loads Files:** Gets HTML, images, etc., from the server.



5. JavaScript on Server

1. **Database Management:** Stores, retrieves, and manages data efficiently through operations like CRUD (Create, Read, Update, Delete).
2. **Authentication:** Verifies user identities to control access to the system, ensuring that users are who they claim to be.
3. **Authorization:** Determines what authenticated users are allowed to do by managing permissions and access controls.
4. **Input Validation:** Checks incoming data for correctness, completeness, and security to prevent malicious data entry and errors.
5. **Session Management:** Tracks user activity across various requests to maintain state and manage user-specific settings.





5. JavaScript on Server

6. **API Management:** Provides and handles interfaces for applications to interact, ensuring smooth data exchange and integration.
7. **Error Handling:** Manages and responds to errors effectively to maintain system stability and provide useful error messages.
8. **Security Measures:** Implements protocols to protect data from unauthorized access and attacks, such as SQL injection and cross-site scripting (XSS).
9. **Data Encryption:** Secures sensitive information by encrypting data stored in databases and during transmission.
10. **Logging and Monitoring:** Keeps records of system activity to diagnose issues and monitor system health and security.

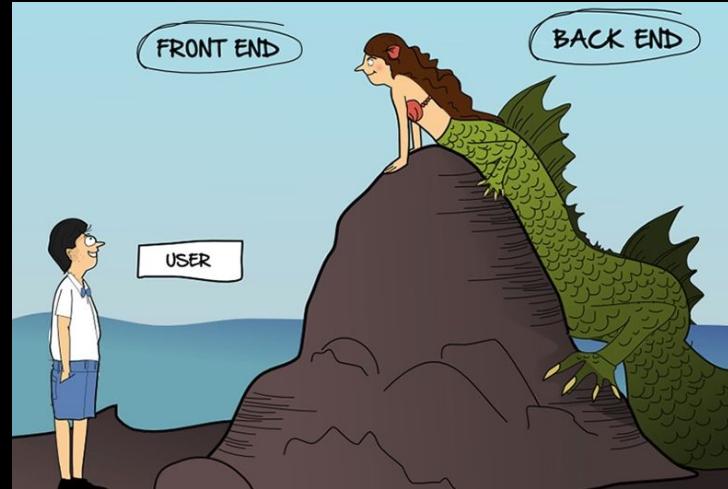
BackEnd / Server-Side





6. Client Code vs Server Code

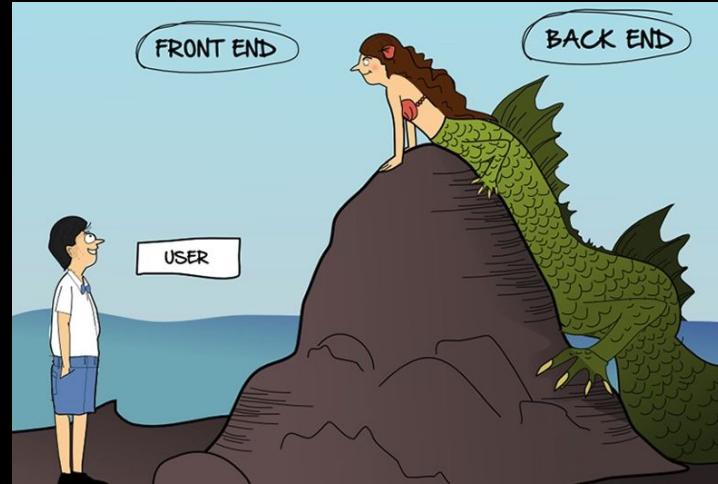
1. User/client can't access server code directly.
2. Client must raise requests for particular APIs to access certain features or data.
3. Environment Access: Server-side JavaScript accesses server features like file systems and databases.
4. Security: Server-side code can handle sensitive operations securely, while client-side code is exposed and must manage security risks.
5. Performance: Heavy computations are better performed on the server to avoid slowing down the client.





6. Client Code vs Server Code

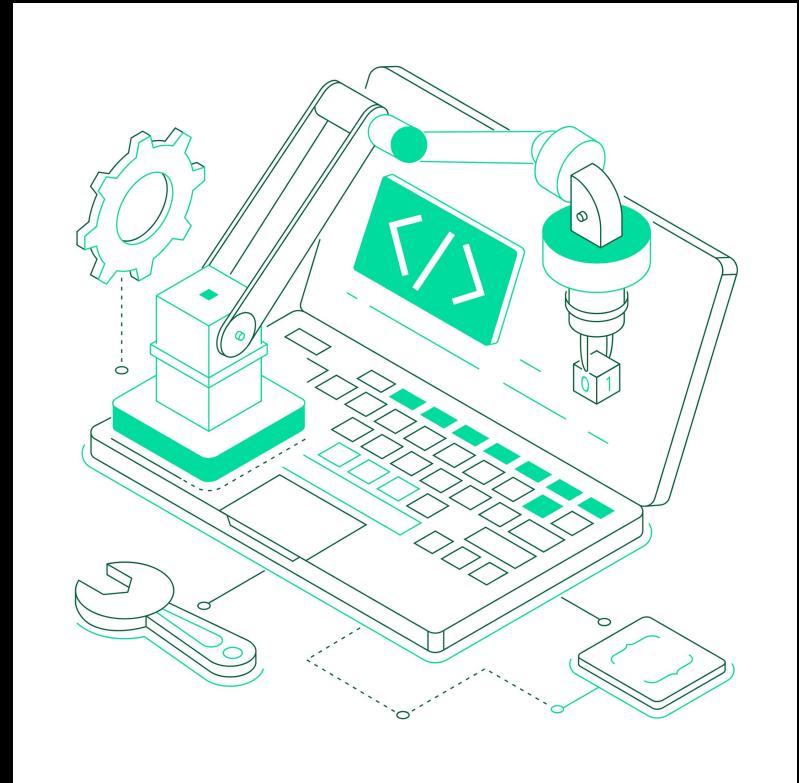
6. **Resource Utilization:** Servers generally offer **more powerful processing capabilities** than client devices.
7. **Data Handling:** Server-side can **directly manage large data sets and database interactions**, unlike client-side JavaScript.
8. **Asynchronous Operations:** Server-side JavaScript is optimized for non-blocking I/O to **efficiently manage multiple requests**.
9. **Session Management:** Servers handle **sessions and user states** more comprehensively.
10. **Scalability:** Server-side code is designed to scale and handle requests from **multiple clients simultaneously**.





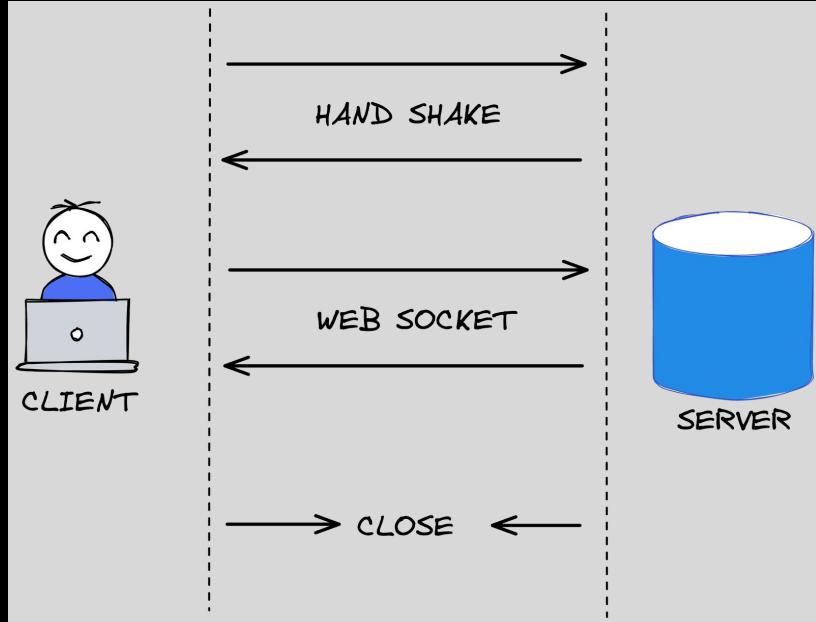
7. Other uses of Node.js

1. **Local Utility Scripts:** Automates tasks and processes files locally, like using shell scripts but with JavaScript.
2. **Internet of Things (IoT):** Develops server-side applications for IoT devices, managing communications and data processing.
3. **Scripting for Automation:** Automates repetitive tasks in software development processes, such as testing and deployment.





7. Other uses of NodeJs



Real-Time Applications: Efficiently manages real-time data applications, such as chat apps and live updates, using WebSockets.



7. Other uses of NodeJs

Apps users love, built with Electron

Thousands of organizations spanning all industries use Electron to build cross-platform software.

The grid contains the following applications:

- 1Password
- Asana
- Discord
- Dropbox
- Figma
- Agora Flat
- Github Desktop
- itch
- Loom
- MongoDB Compass
- Notion
- Obsidian
- Polpane
- Postman
- Signal
- Skype
- Slack
- Splice
- Microsoft Teams
- Tidal
- Trello
- Twitch
- VS Code
- WordPress Desktop

Desktop Applications: Creates **cross-platform desktop applications** using frameworks like **Electron**.



7. Other uses of NodeJs

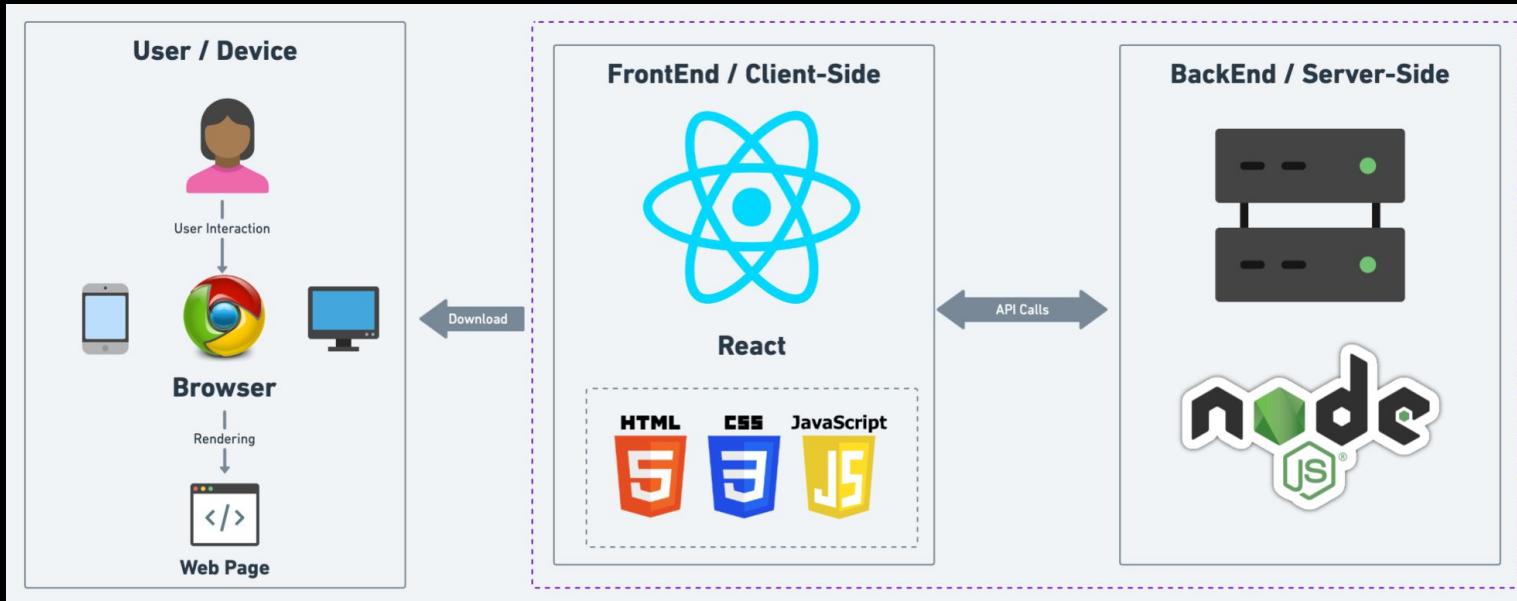
Build Tools: Powers build processes for front-end technologies using tools like:

- Webpack
- Grunt
- Gulp
- Browserify
- Brunch
- Yeoman





8. Server architecture with NodeJs



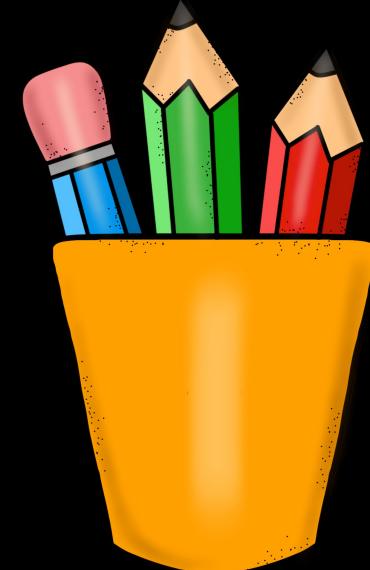
Nodejs server will:

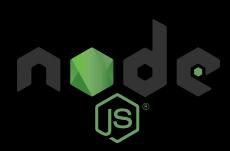
1. Create server and **listen to incoming requests**
2. **Business logic:** validation, connect to db, actual processing of data
3. Return response **HTML, JSON, CSS, JS**



Revision

1. Pre-requisites
2. What is NodeJS
3. NodeJs Features
4. JavaScript on Client
5. JavaScript on Server
6. Client Code vs Server Code
7. Other uses of NodeJs
8. Server architecture with NodeJs







2. Installation of NodeJS

1. What is IDE
2. Need of IDE
3. MAC Setup
 - Install latest Node & VsCode
4. Windows Setup
 - Install latest Node & VsCode
5. Linux Setup
 - Install latest Node & VsCode
6. VsCode (Extensions and Settings)
7. Executing first .js file
8. What is REPL
9. Executing Code via REPL





2.1 What is IDE

1. IDE stands for **Integrated Development Environment**.
2. Software suite that consolidates basic tools required for **software development**.
3. Central hub for **coding**, finding problems, and testing.
4. Designed to improve **developer efficiency**.





2.2 Need of IDE

1. Streamlines development.
2. Increases productivity.
3. Simplifies complex tasks.
4. Offers a unified workspace.
5. IDE Features
 1. Code Autocomplete
 2. Syntax Highlighting
 3. Version Control
 4. Error Checking

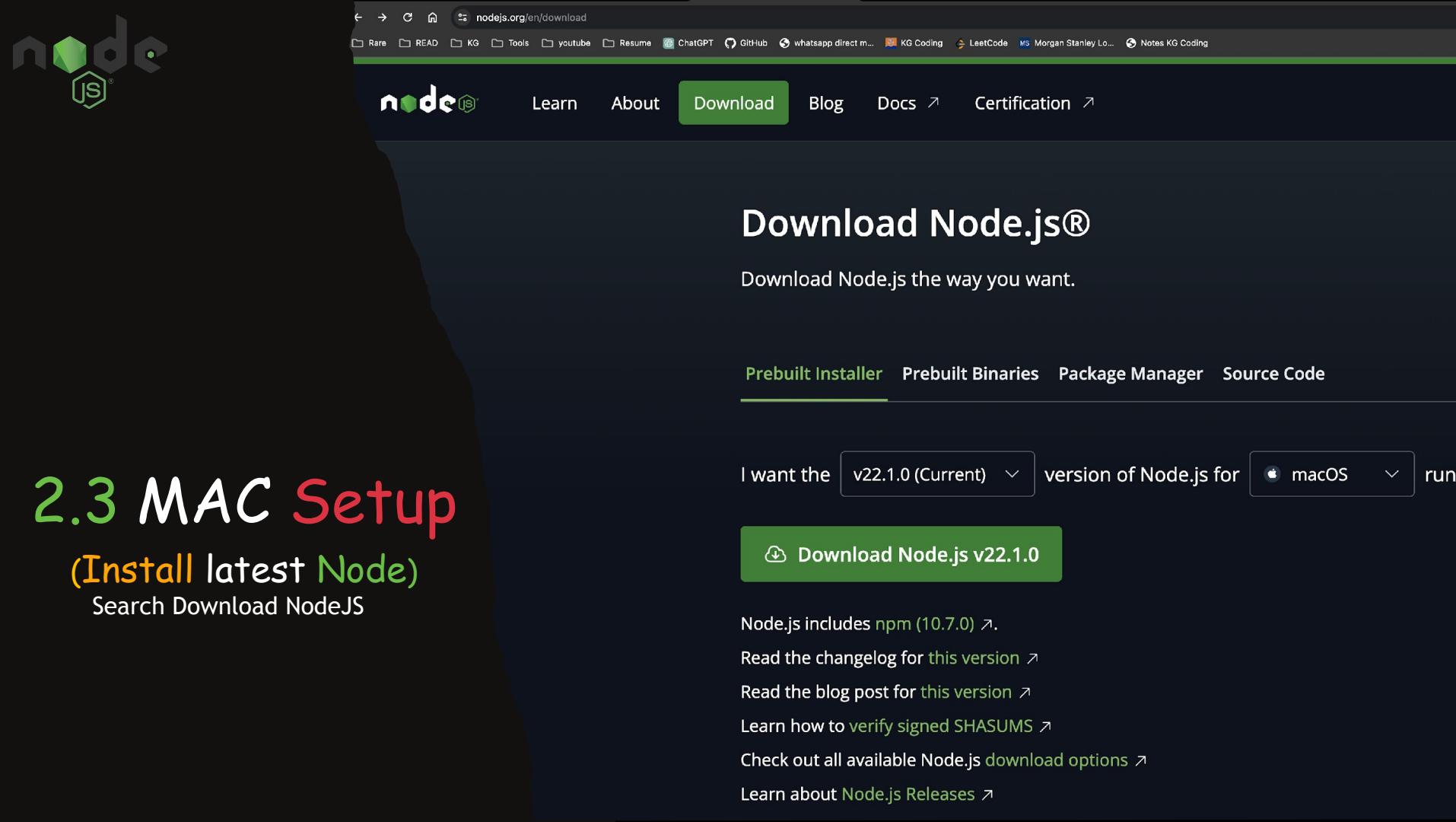
```
MainActivity.kt
```

```
@Composable
fun MessageCard(msg: Message) {
    Row(modifier = Modifier.padding(all = 8.dp)) {
        Image(
            painter = painterResource(R.drawable.android_studio_logo),
            contentDescription = "Profile Picture",
            modifier = Modifier
                .size(45.dp)
        )
        Spacer(modifier = Modifier.width(8.dp))
        Column (Modifier
            .background(color = Color.White)) {
            Text(text = msg.author, color = Color.Black)
            Spacer(modifier = Modifier.height(1.dp))
            Text(text = msg.body, color = Color.Black)
        }
    }
}
```



2.3 MAC Setup





2.3 MAC Setup

(Install latest Node)

Search Download NodeJS

nodejs.org/en/download

Rare READ KG Tools youtube Resume ChatGPT GitHub whatsapp direct m... KG Coding LeetCode Morgan Stanley Lo... Notes KG Coding

node Learn About Download Blog Docs Certification

Download Node.js®

Download Node.js the way you want.

Prebuilt Installer Prebuilt Binaries Package Manager Source Code

I want the v22.1.0 (Current) version of Node.js for macOS

Download Node.js v22.1.0

Node.js includes npm (10.7.0).

Read the changelog for this version

Read the blog post for this version

Learn how to verify signed SHASUMS

Check out all available Node.js download options

Learn about Node.js Releases



2.3 MAC Setup

(Install VsCode)



Search VS Code on Google

Code editing.
Redefined.

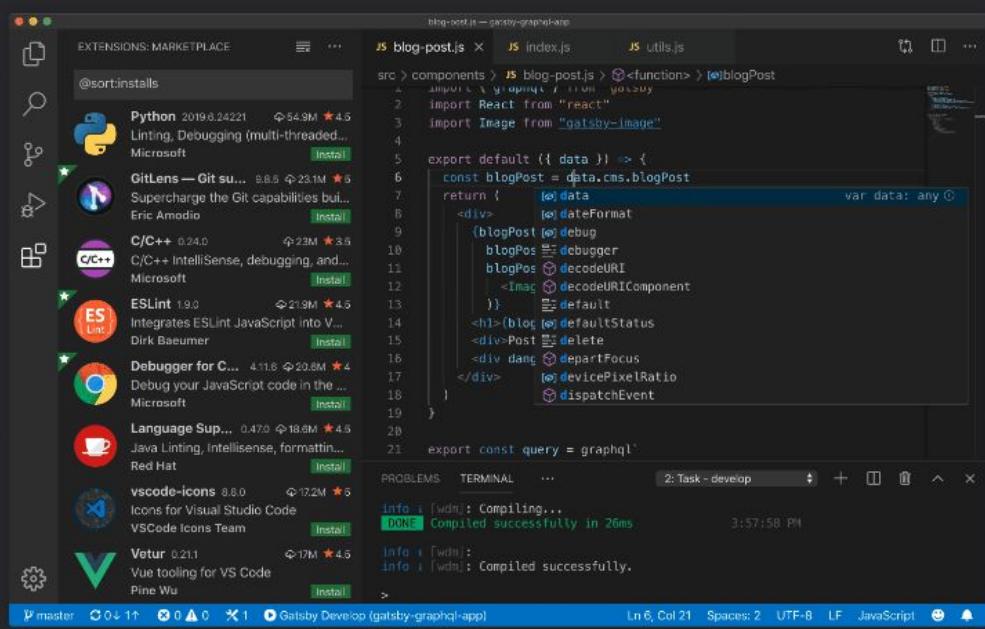
Free. Built on open source. Runs everywhere.

[Download Mac Universal](#)

Stable Build

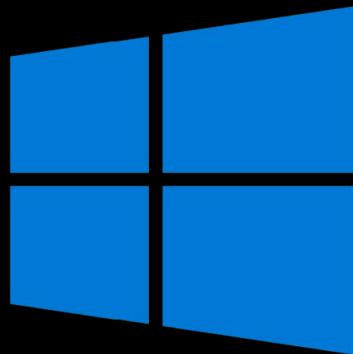
Web, Insiders edition, or other platforms

By using VS Code, you agree to its
[license and privacy statement](#).





2.4 Windows Setup



Windows



nodejs.org/en/download

KG Tools youtube Resume ChatGPT GitHub whatsapp direct m... KG Coding LeetCode Morgan Stanley Lo... Notes KG Coding

Learn About Download Blog Docs ↗ Certification ↗

Download Node.js®

Download Node.js the way you want.

Prebuilt Installer Prebuilt Binaries Package Manager Source Code

I want the v22.1.0 (Current) version of Node.js for Windows running x64

[Download Node.js v22.1.0](#)

Node.js includes npm (10.7.0).

Read the changelog for this version ↗

Read the blog post for this version ↗

Learn how to verify signed SHASUMS ↗

Check out all available Node.js download options ↗

Learn about Node.js Releases ↗

2.4 Windows Setup

(Install latest Node)

Search Download NodeJS



2.4 Windows Setup (Install VsCode)



Search VS Code on Google

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows

Stable Build

By using VS Code, you agree to its license and privacy statement.

```
src > JS serviceWorker.js > register > window.addEventListener('load') callback
    checkValidServiceWorker(swUrl, config);

    // Add some additional logging to localhost, p...
    // service worker/PWA documentation.
    navigator.serviceWorker.ready.then(() => {

        product
        productSub
        removeSiteSpecificTrackingException
        removeWebWideTrackingException
        requestMediaKeySystemAccess
        sendBeacon

        serviceWorker (property) Navigator.serviceWorker...
        storage
        storeSiteSpecificTrackingException
        storeWebWideTrackingException
    ) userAgent
    vendor

function registerValidSW(swUrl, config) {
    navigator.serviceWorker
        .register(swUrl)
        .then(registration => {

```

TERMINAL ... 1:node + ^ x

You can now view `create-react-app` in the browser.

Local: `http://localhost:3000/`
On Your Network: `http://10.211.55.3:3000/`

Note that the development build is not optimized.



2.5 Linux Setup



Linux



Node.js — Download Node X nodejs for linux - Google X Node.js — Download Node X +

https://nodejs.org/en/download/package-manager/ Node.js v22 is now available! ↗

Learn About Download Blog Docs ↗ Certification ↗

Download Node.js®

Download Node.js the way you want.

Prebuilt Installer Prebuilt Binaries **Package Manager** Source Code

Install Node.js v22.1.0 (Current) on Linux using NVM

```
1 # installs NVM (Node Version Manager)
2 curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | b
3
4 # download and install Node.js
5 nvm install 22
6
7 # verifies the right Node.js version is in the environment
8 node -v # should print 'v22.1.0'
9
10 # verifies the right NPM version is in the environment
11 npm -v # should print '10.7.0'
```

Bash Copy

Please ensure you have the right package manager installed before running a script. Package managers and their installation scripts are not maintained by the Node.js project.



2.5 Linux Setup

(Install VsCode)

Search VS Code on Google



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows

Windows 10, 11



↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE



↓ Mac

macOS 10.15+

| | | |
|------------------|-----|-------|
| User Installer | x64 | Arm64 |
| System Installer | x64 | Arm64 |
| .zip | x64 | Arm64 |
| CLI | x64 | Arm64 |

| | | | |
|---------|------------|-------|-------|
| .deb | x64 | Arm32 | Arm64 |
| .rpm | x64 | Arm32 | Arm64 |
| .tar.gz | x64 | Arm32 | Arm64 |
| Snap | Snap Store | | |
| CLI | x64 | Arm32 | Arm64 |

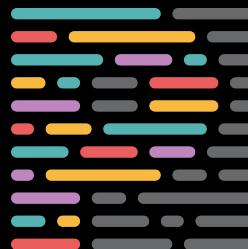
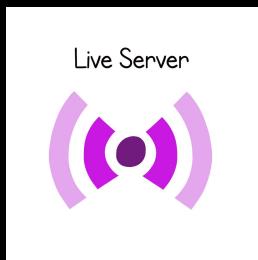
| | | | |
|------|------------|---------------|-----------|
| .zip | Intel chip | Apple silicon | Universal |
| CLI | Intel chip | Apple silicon | |



2.6 VsCode

(Extensions and Settings)

1. Prettier (Format on Save)
2. Line Wrap
3. Tab Size from 4 to 2





2.7 Executing first .js file

```
1 const fs = ...require('fs');
2
3 // Define two variables
4 let a = 10;
5 let b = 5;
6
7 // Basic arithmetic operations
8 let sum = a + b;
9 let product = a * b;
10
11 // Prepare data to write
12 let data = `Sum: ${sum}\nProduct: ${product}`;
13 console.log(data);
14
15 // Write data to a local file
16 fs.writeFile('output.txt', data, (err) => {
17   |   if (err) throw err;
18   |   console.log('Data written to file');
19 });
```

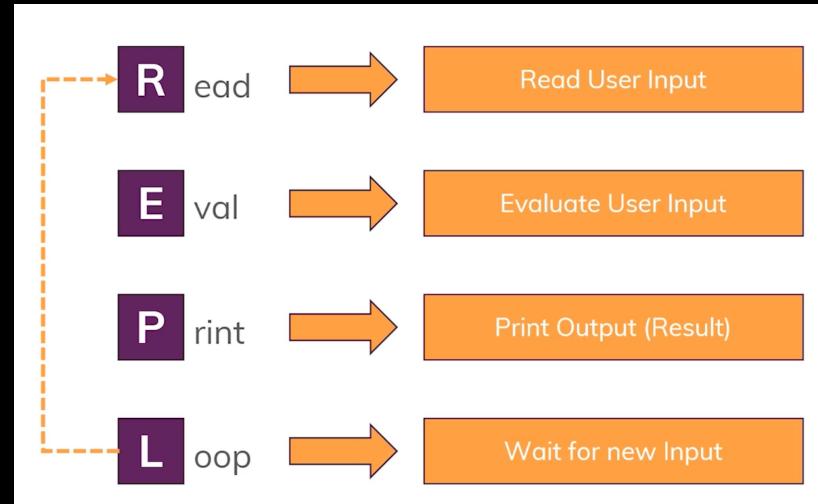
```
prashantjain@Mac-mini Desktop % node test.js
Sum: 15
Product: 50
Data written to file
```

1. Streamlines Node Command: Use `node filename.js` to execute a JavaScript file in the Node.js environment.
2. Require Syntax: Use `require('module')` to include built-in or external modules, or other JavaScript files in your code.
3. Modular Code: `require` helps organize code into reusable modules, separating concerns and improving maintainability.
4. Caching: Modules loaded with `require` are cached, meaning the file is executed only once even if included multiple times.



2.8 What is REPL

1. Streamlines Interactive Shell: Executes JavaScript code interactively.
2. Quick Testing: Ideal for testing and debugging code snippets on the fly.
3. Built-in Help: Offers help commands via .help.
4. Session Management: Supports saving (.save) and loading (.load) code sessions.
5. Node.js API Access: Provides direct access to Node.js APIs for experimentation.
6. Customizable: Allows customization of prompt and behaviour settings.





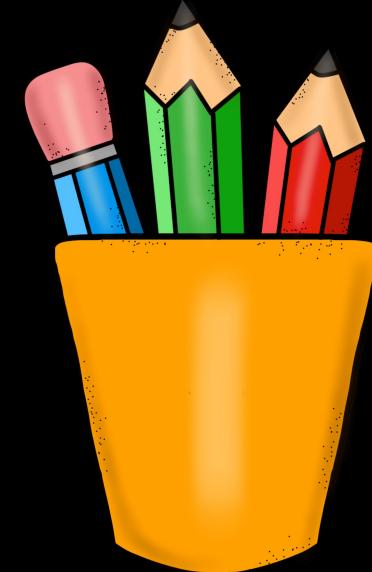
2.9 Executing Code via REPL

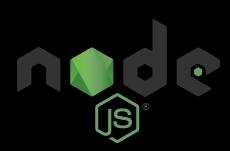
```
prashantjain@Mac-mini Desktop % node
Welcome to Node.js v20.9.0.
Type ".help" for more information.
> 5 + 6
11
> console.log('KG Coding is the best');
KG Coding is the best
undefined
> fs.writeFile('output.txt', 'Writing to file', (err) => {
...     if (err) throw err;
...     console.log('Data written to file');
... });
undefined
> Data written to file
```



Revision

1. What is IDE
2. Need of IDE
3. MAC Setup
 - Install latest Node & VsCode
4. Windows Setup
 - Install latest Node & VsCode
5. Linux Setup
 - Install latest Node & VsCode
6. VsCode (Extensions and Settings)
7. Executing first .js file
8. What is REPL
9. Executing Code via REPL







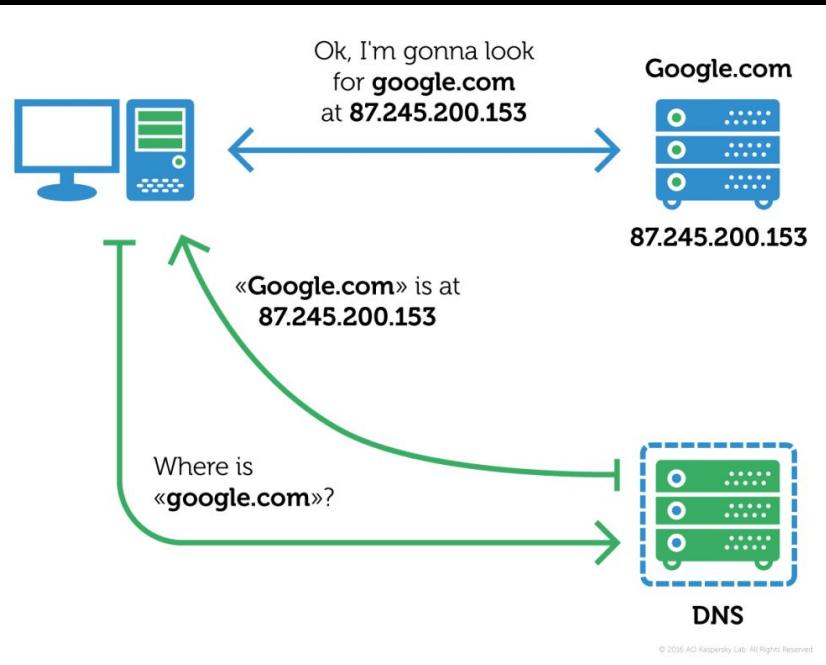
3. First Node Server

1. How DNS Works?
2. How Web Works?
3. What are Protocols?
4. Node Core Modules
5. Require Keyword
6. Creating first Node Server





3.1 How DNS Works?

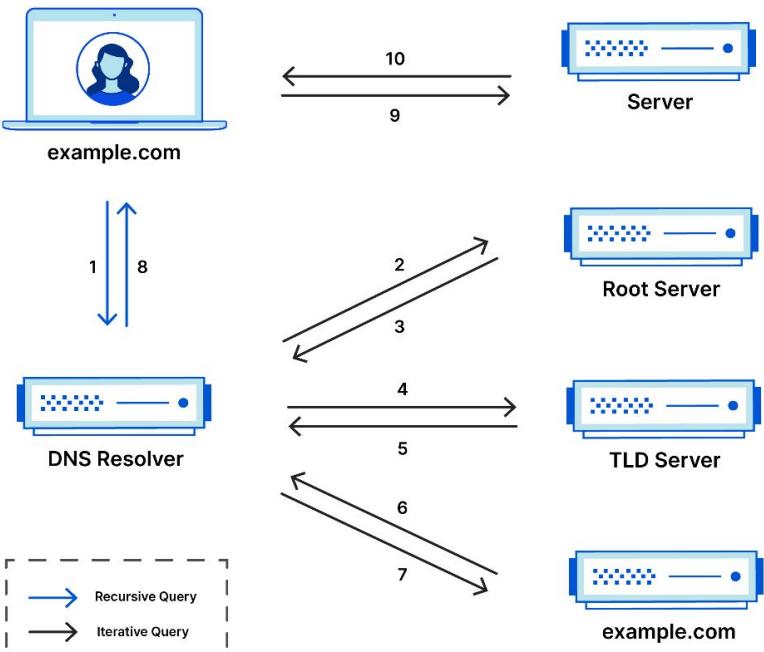


1. **Domain Name Entry:** User types a domain (e.g., **www.example.com**) into the browser.
2. **DNS Query:** The browser **sends a DNS query** to resolve the domain into an IP address.
3. **DNS Server:** **Provides the correct IP address** for the domain.
4. **Browser Connects:** The browser uses the IP to connect to the web server and loads the website.



3.1 How DNS Actually Works?

Complete DNS Lookup and Webpage Query

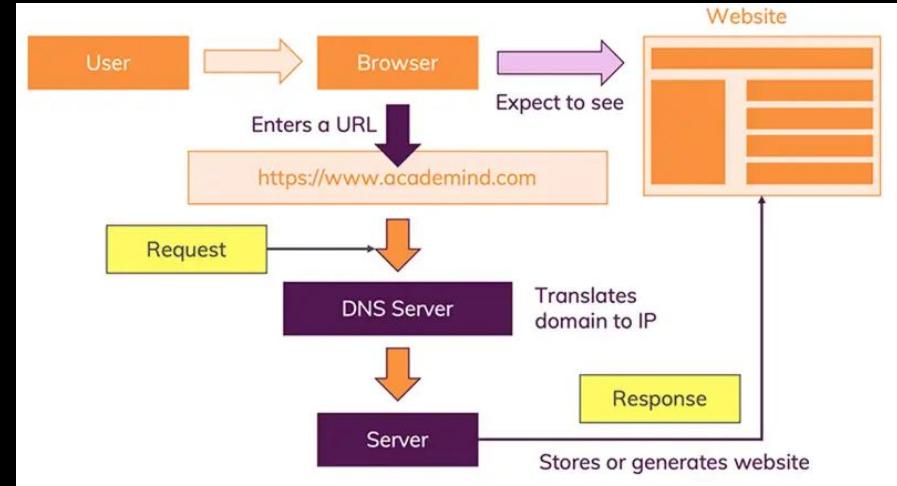


1. **Root DNS:** Acts as the starting point for DNS resolution. It directs queries to the correct TLD server (e.g., .com, .org).
2. **TLD (Top-Level Domain) DNS:** Handles queries for specific top-level domains (e.g., .com, .net) and directs them to the authoritative DNS server (e.g., Verisign for .com, PIR for .org)
3. **Authoritative DNS:** Contains the actual IP address of the domain and answers DNS queries with this information. (e.g., Cloudflare, Google DNS).



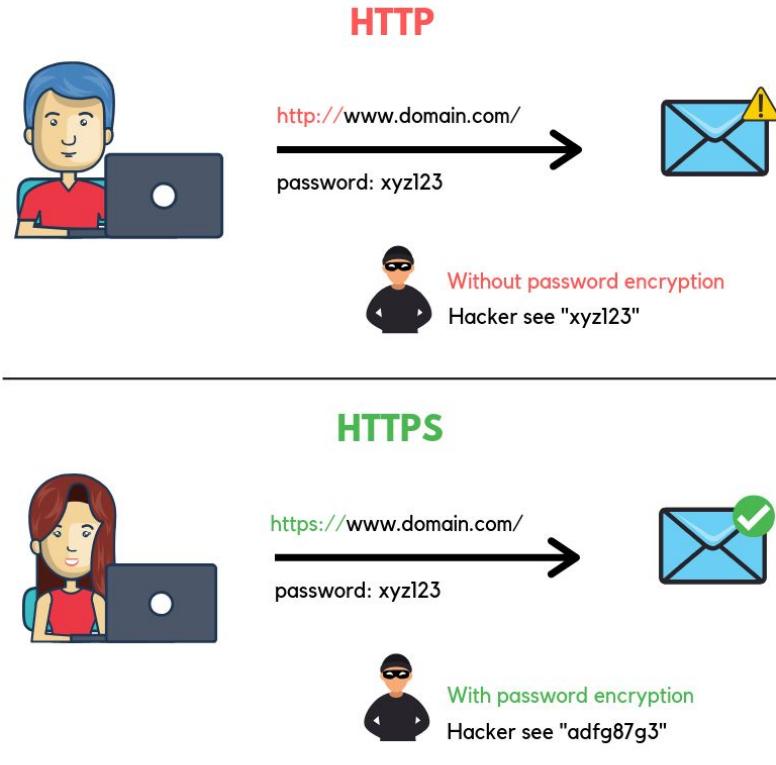
3.2 How Web Works?

1. **Client Request Initiation:** The client (browser) initiates a network call by entering a URL.
2. **DNS Resolution:** The browser contacts a DNS server to get the IP address of the domain.
3. **TCP Connection:** The browser establishes a TCP connection with the server's IP address.
4. **HTTP Request:** The browser sends an HTTP request to the server.
5. **Server Processing:** The server processes the request and prepares a response.
6. **HTTP Response:** The server sends an HTTP response back to the client.
7. **Network Transmission:** The response travels back to the client over the network.
8. **Client Receives Response:** The browser receives and interprets the response.
9. **Rendering:** The browser renders the content of the response and displays it to the user.





3.3 What are Protocols?



Http (HyperText Transfer Protocol):

- Facilitates **communication** between a web browser and a server to transfer web pages.
- Sends data in **plain text (no encryption)**.
- Used for **basic website browsing** without security.

HTTPS (HyperText Transfer Protocol Secure):

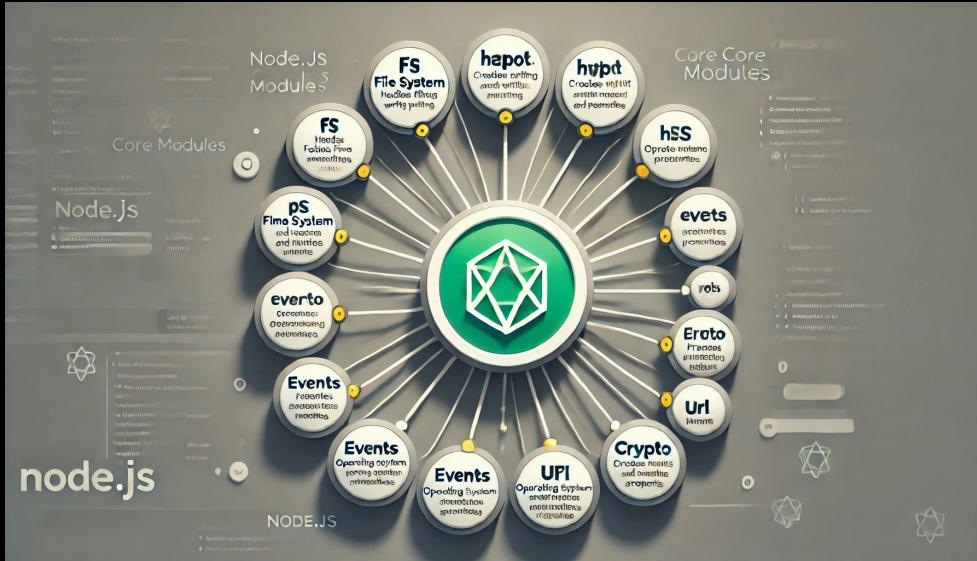
- **Secure version of HTTP**, encrypts data for secure communication.
- Uses **SSL/TLS to encrypt data**.
- Used in **online banking, e-commerce**.

TCP (Transmission Control Protocol):

- **Ensures reliable, ordered, and error-checked data delivery** over the internet.
- Establishes a connection before data is transferred.



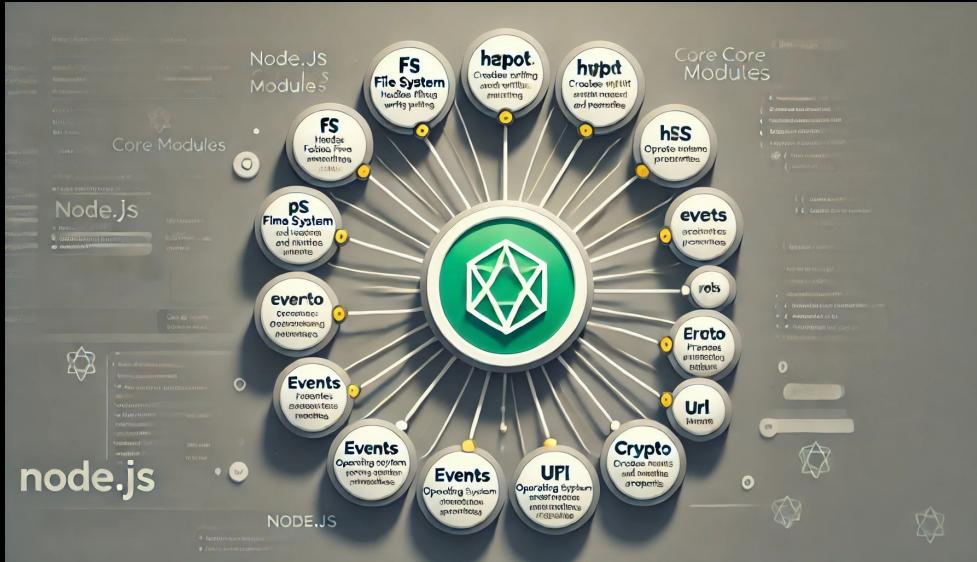
3.4 Node Core Modules



1. **Built-in:** Core modules are included with Node.js installation.
2. **No Installation Needed:** Directly available for use without npm install.
3. **Performance:** Highly optimized for performance.



3.4 Node Core Modules



1. **Built-in:** Core modules are included with Node.js installation.
2. **No Installation Needed:** Directly available for use without npm install.
3. **Performance:** Highly optimized for performance.



3.4 Node Core Modules

1. `fs` (File System): Handles file operations like reading and writing files.
2. `http`: Creates HTTP servers and makes HTTP requests.
3. `https`: Launch a SSL Server.
4. `path`: Provides utilities for handling and transforming file
5. `path.os`: Provides operating system-related utility methods and properties.
6. `events`: Handles events and event-driven programming.
7. `crypto`: Provides cryptographic functionalities like hashing and encryption.
8. `url`: Parses and formats URL strings.



3.5 Require Keyword

1. **Purpose:** Imports modules in `Node.js`.
2. **Caching:** Modules are `cached` after the first `require` call.
3. `.js` is added automatically and is not needed to at the end of module name.
4. **Path Resolution:** `Node.js` searches for modules in `core`, `node_modules`, and file paths.

Syntax:

```
const moduleName = require('module');
```

```
// Load the built-in http module  
const http = ...require('http');
```

```
// Load the third party express module  
const express = require('express');
```

```
// Load the custom myModule module  
const myModule = require('./myModule');
```

3.6 Creating first Node Server

```
1 // Simple Node.js server
2 const http = require('http');
3
4 function requestListener(req, res) {
5   |   console.log(req);
6 }
7
8 http.createServer(requestListener);
```

3.6 Creating first Node Server

```
1 // Simple Node.js server
2 const http = require('http');
3
4 http.createServer(function (req, res) {
5   console.log(req);
6 });
```

3.6 Creating first Node Server

```
1 // Simple Node.js server
2 const http = require('http');
3
4 http.createServer((req, res) => {
5   | console.log(req);
6 });


```

Run the code with:

node app.js



3.6 Creating first Node Server

```
1 // Simple Node.js server
2 const http = require('http');
3
4 const server = http.createServer((req, res) => {
5   console.log(req);
6 });
7
8 server.listen(3000);
```

```
insecureHTTPParser: undefined,
requestTimeout: 300000,
headersTimeout: 60000,
keepAliveTimeout: 5000,
connectionsCheckingInterval: 30000,
requireHostHeader: true,
joinDuplicateHeaders: undefined,
rejectNonStandardBodyWrites: false,
_events: [Object: null prototype],
_eventsCount: 3,
_maxListeners: undefined,
_connections: 2,
_handle: [TCP],
_usingWorkers: false,
_workers: [],
_unref: false,
_listeningId: 2,
allowHalfOpen: true,
pauseOnConnect: false,
noDelay: true,
keepAlive: false,
keepAliveInitialDelay: 0,
highWaterMark: 65536,
httpAllowHalfOpen: false,
timeout: 0,
maxHeadersCount: null,
```

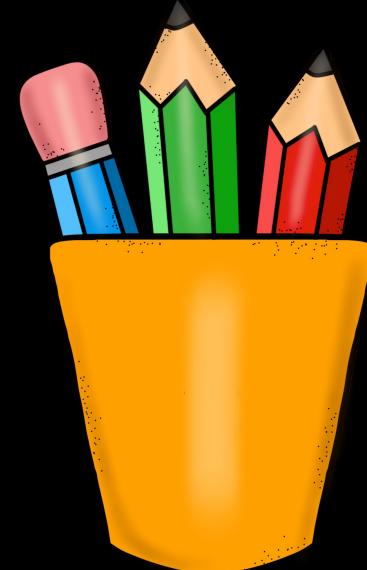
3.6 Creating first Node Server

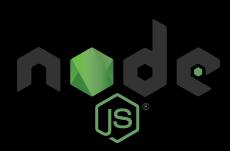
```
1 // Simple NodeJS server
2 const http = require('http');
3
4 const server = http.createServer((req, res) => {
5   console.log(req);
6 });
7
8 const PORT = 3000;
9 server.listen(PORT, () => {
10   console.log(`Server running at http://localhost:${PORT}/`);
11});
```



Revision

1. How DNS Works?
2. How Web Works?
3. What are Protocols?
4. Node Core Modules
5. Require Keyword
6. Creating first Node Server





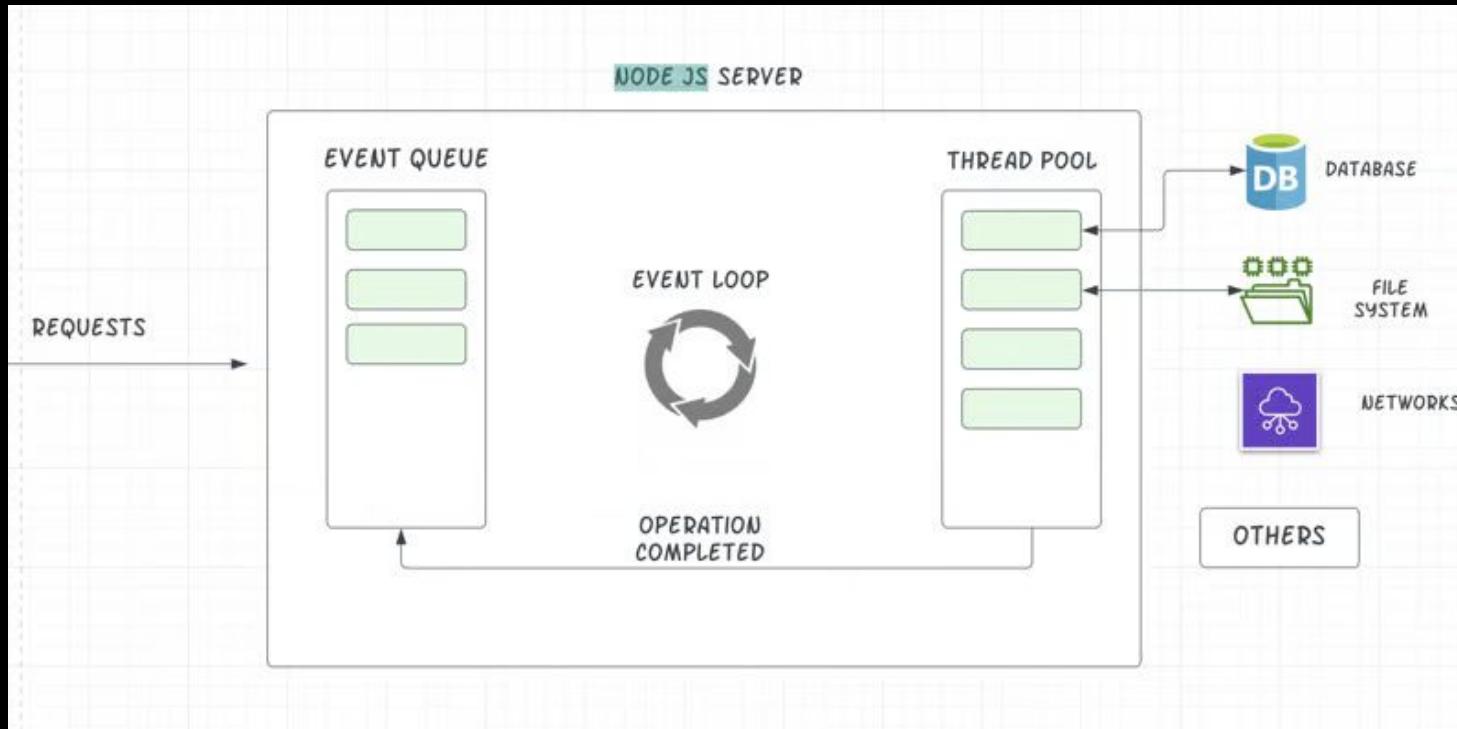


4. Request & Response

1. Node Lifecycle & Event Loop
2. How to exit Event Loop
3. Understand Request Object
4. Sending Response
5. Routing Requests
6. Taking User Input
7. Redirecting Requests



node 4.1 Node Lifecycle & Event Loop





4.2 How to exit Event Loop

```
1 // Simple Node.js server
2 const http = require('http');
3
4 const server = http.createServer((req, res) => {
5   console.log(req);
6   process.exit(); // Stops event loop
7 });
8
9 const PORT = 3000;
10 server.listen(PORT, () => {
11   console.log(`Server running at http://localhost:${PORT}/`);
12 });
```

4.3 Understand Request Object

```
[Symbol(kHeaders)]: {
  host: 'localhost:3000',
  connection: 'keep-alive',
  'cache-control': 'max-age=0',
  'sec-ch-ua': '"Chromium";v="128", "Not;A=Brand";v="24", "Google Chrome";v="128"',
  'sec-ch-ua-mobile': '?0',
  'sec-ch-ua-platform': '"macOS"',
  'upgrade-insecure-requests': '1',
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.6482.129 Safari/537.36',
  accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7',
  'sec-fetch-site': 'none',
  'sec-fetch-mode': 'navigate',
  'sec-fetch-user': '?1',
  'sec-fetch-dest': 'document',
  'accept-encoding': 'gzip, deflate, br, zstd',
  'accept-language': 'en-US,en-IN;q=0.9,en;q=0.8,hi-IN;q=0.7,hi;q=0.6',
  cookie: 'token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImM3M2Y0MzNjLTFlYzYtNDUyUSrMLqbP1uy5bTjnJEQHXc1c',
},
[Symbol(kHeadersCount)]: 32,
[Symbol(kTrailers)]: null,
[Symbol(kTrailersCount)]: 0
}
```



4.3 Understand Request Object

```
// Simple NodeJS server
const http = require('http');

const server = http.createServer((req, res) => {
|   console.log(req.url, req.method, req.headers);
});

const PORT = 3000;
server.listen(PORT, () => {
|   console.log(`Server running at http://localhost:${PORT}/`);
});
```

Use the browser to access:

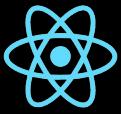
<http://localhost:3000/>

<http://localhost:3000/products>



4.4 Sending Response

```
1 // Simple NodeJS server
2 const http = require('http');
3
4 const server = http.createServer((req, res) => {
5   //res.setHeader('Content-Type', 'json');
6   res.setHeader('Content-Type', 'text/html');
7   res.write('<html>');
8   res.write('<head><title>Complete Coding</title></head>');
9   res.write('<body><h1>Like / Share / Subscribe</h1></body>');
10  res.write('</html>');
11  res.end();
12 });
13
14 const PORT = 3000;
15 server.listen(PORT, () => {
16   console.log(`Server running at http://localhost:${PORT}/`);
17 })
```



React

4.5 Routing Requests

```
const server = http.createServer((req, res) => {
  res.setHeader('Content-Type', 'text/html');
  res.write('<html>');
  res.write('<head><title>Complete Coding</title></head>');
  if (req.url === '/') {
    res.write('<h1>Welcome to Home page</h1>');
    res.end();
  } else if (req.url.toLowerCase() === '/products') {
    res.write('<h1>Products</h1>');
    res.end();
  }
  res.write('<body><h1>Like / Share / Subscribe</h1></body>');
  res.write('</html>');
  res.end();
});
```

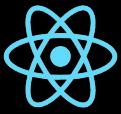


4.5 Routing Requests

```
⑥ prashantjain@Prashants-Mac-mini node % node app.js
Server running at http://localhost:3000/
node:_http_outgoing:699
    throw new ERR_HTTP_HEADERS_SENT('set');
^
```

```
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
at ServerResponse.setHeader (node:_http_outgoing:699:11)
at Server.<anonymous> (/Users/prashantjain/workspace/Test Project/node/app.js:14:7)
at Server.emit (node:events:520:28)
at parserOnIncoming (node:_http_server:1146:12)
at HTTPParser.parserOnHeadersComplete (node:_http_common:118:17) {
  code: 'ERR_HTTP_HEADERS_SENT'
}
```

Node.js v22.5.1



React

4.5 Routing Requests

```
const server = http.createServer((req, res) => {
  res.setHeader('Content-Type', 'text/html');
  res.write('<html>');
  res.write('<head><title>Complete Coding</title></head>');
  if (req.url === '/') {
    res.write('<h1>Welcome to Home page</h1>');
    return res.end();
  } else if (req.url.toLowerCase() === '/products') {
    res.write('<h1>Products</h1>');
    return res.end();
  }
  res.write('<body><h1>Like / Share / Subscribe</h1></body>');
  res.write('</html>');
  return res.end();
});
```



React

4.6 Taking User Input

```
if (req.url === '/') {  
  res.write('<h1>Welcome to Home page</h1>');  
  res.write('<form action="/submit-details" method="POST">');  
  res.write('<input type="text" id="name" name="name" placeholder="Enter your  
name"><br><br>');  
  res.write('<label for="gender">Gender:</label>');  
  res.write('<input type="radio" id="male" name="gender" value="male">');  
  res.write('<label for="male">Male</label>');  
  res.write('<input type="radio" id="female" name="gender" value="female">');  
  res.write('<label for="female">Female</label><br><br>');  
  res.write('<button type="submit">Submit</button>');  
  res.write('</form>');  
  return res.end();  
}
```



4.7 Redirecting Requests

```
} else if (req.method == 'POST' &&
           req.url.toLowerCase() === '/submit-details') {
  fs.writeFileSync('user-details.txt', 'Prashant Jain');
  res.statusCode = 302;
  res.setHeader('Location', '/');
  return res.end();
}
```

```
node > ≈ user-details.txt
```

```
1 Prashant Jain
```



React

Practise Set

Create a page that shows a navigation bar of Myntra with the following links:

- A. Home
- B. Men
- C. Women
- D. Kids
- E. Cart

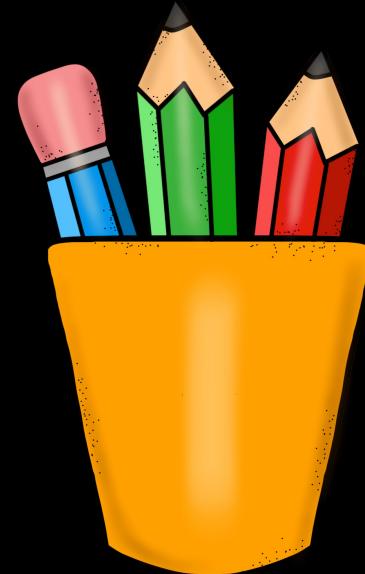
Clicking on each link **page** should navigate to that **page** and a welcome to section **text** is shown there.

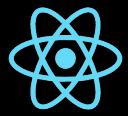




Revision

1. Node Lifecycle & Event Loop
2. How to exit Event Loop
3. Understand Request Object
4. Sending Response
5. Routing Requests
6. Taking User Input
7. Redirecting Requests





React



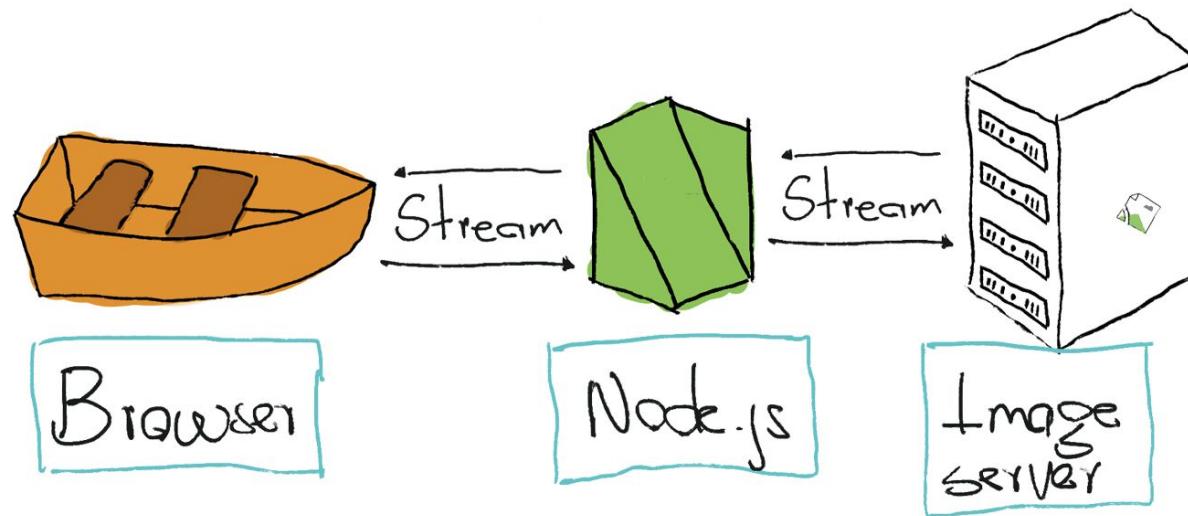
5. Parsing Request

1. Streams
2. Chunks
3. Buffers
4. Reading Chunk
5. Buffering Chunks
6. Parsing Request
7. Using Modules



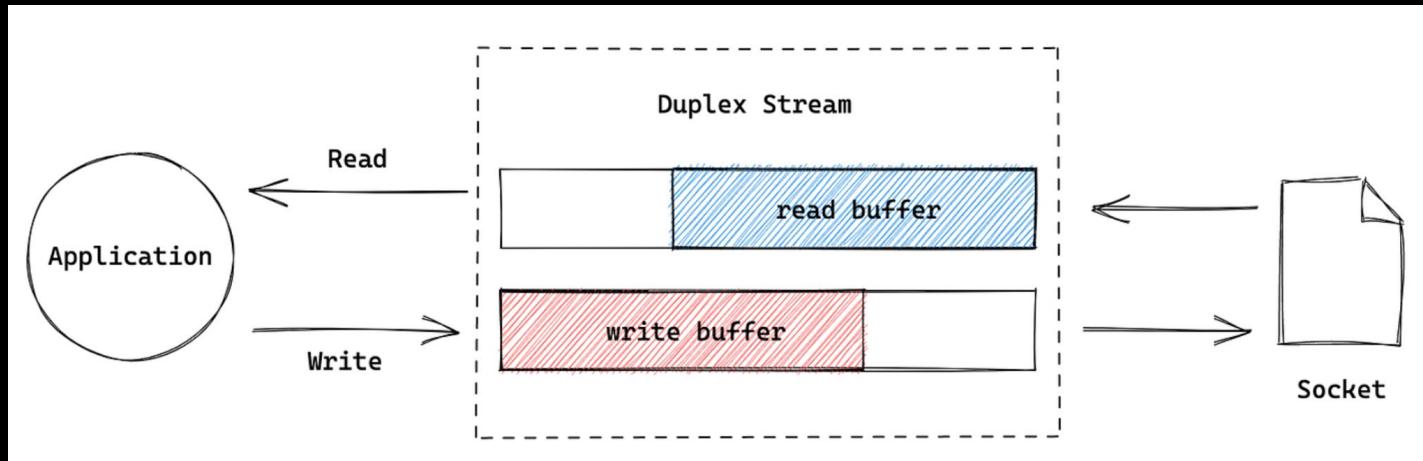


5.1 Streams

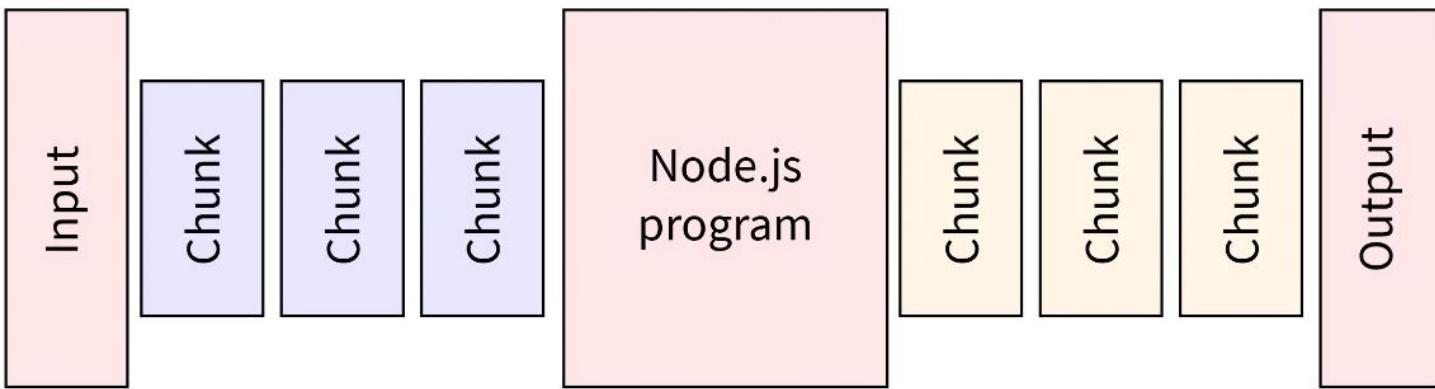




5.1 Streams

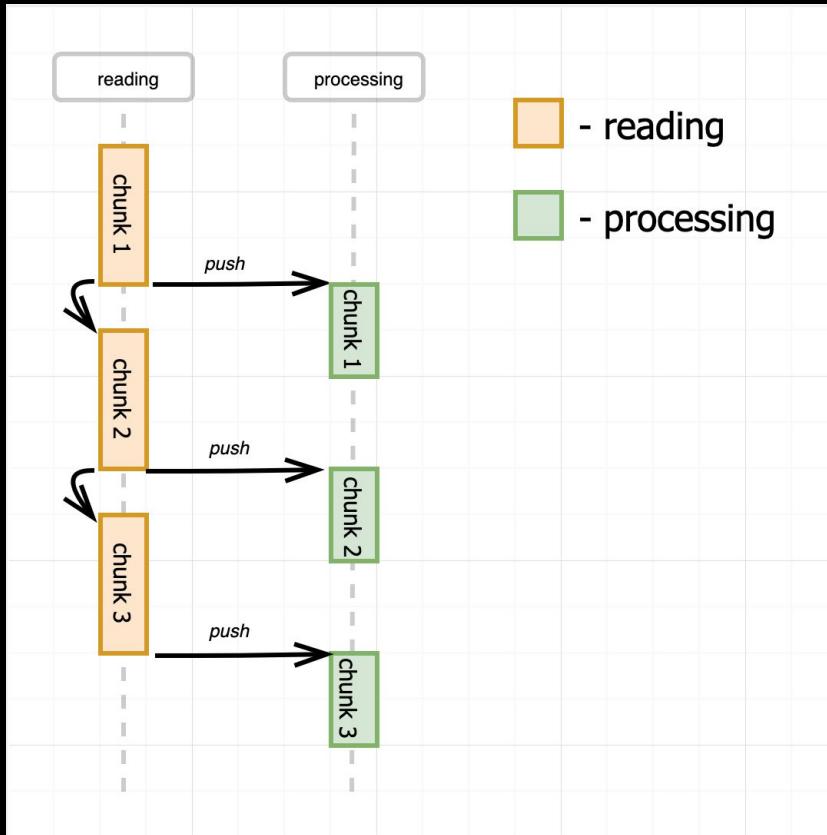


5.2 Chunks



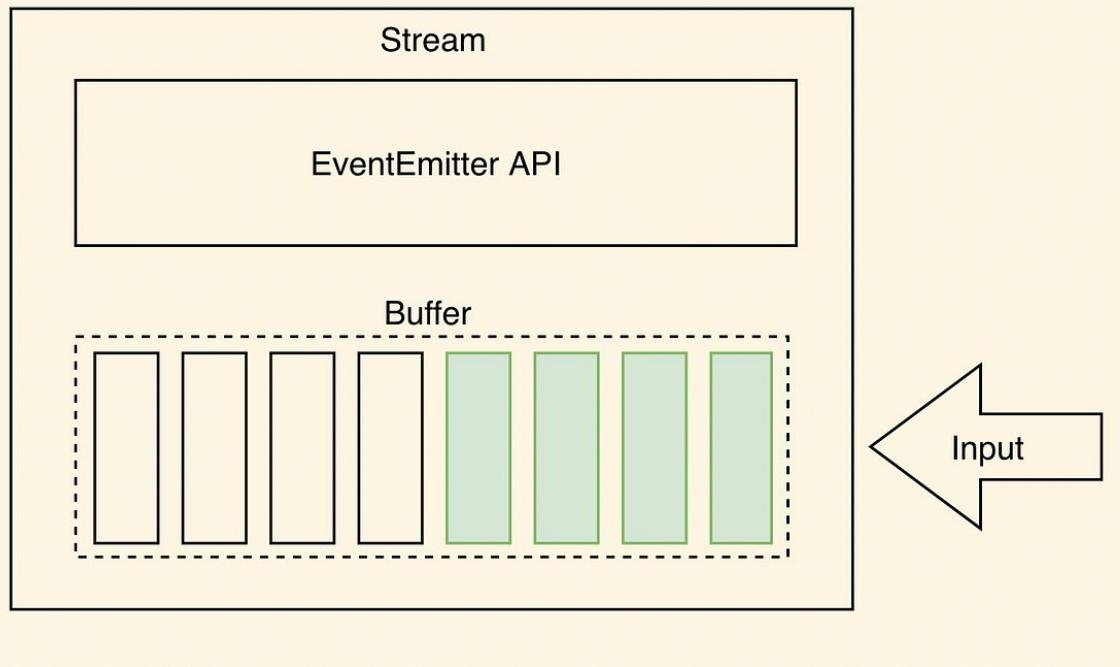


5.2 Chunks





5.3 Buffers

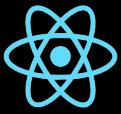




5.4 Reading Chunk

```
} else if (
  req.method == "POST" &&
  req.url.toLowerCase() === "/submit-details"
) {
  req.on("data", (chunk) => {
    console.log(chunk);
  });
  fs.writeFileSync("user-details.txt", "Prashant Jain");
  res.setHeader("Location", "/");
  res.statusCode = 302;
  return res.end();
}
```

```
prashantjain@Prashants-Mac-mini node % node app.js
Server running at http://localhost:3000/
<Buffer 6e 61 6d 65 3d 50 72 61 73 68 61 6e 74 26 67 65 6e 64 65 72 3d 6d 61 6c 65>
```



React

5.5 Buffering Chunks

```
const body = [];
req.on("data", (chunk) => {
  console.log(chunk);
  body.push(chunk);
});
req.on("end", () => {
  const parsedBody = Buffer.concat(body).toString();
  console.log(parsedBody);
});
fs.writeFileSync("user-details.txt", "Prashant Jain");
```

```
.prashantjain@Prashants-Mac-mini % node app.js
Server running at http://localhost:3000/
<Buffer 6e 61 6d 65 3d 50 72 61 73 68 61 6e 74 26 67 65 6e 64 65 72 3d 6d 61 6c 65>
name=Prashant&gender=male
%
```



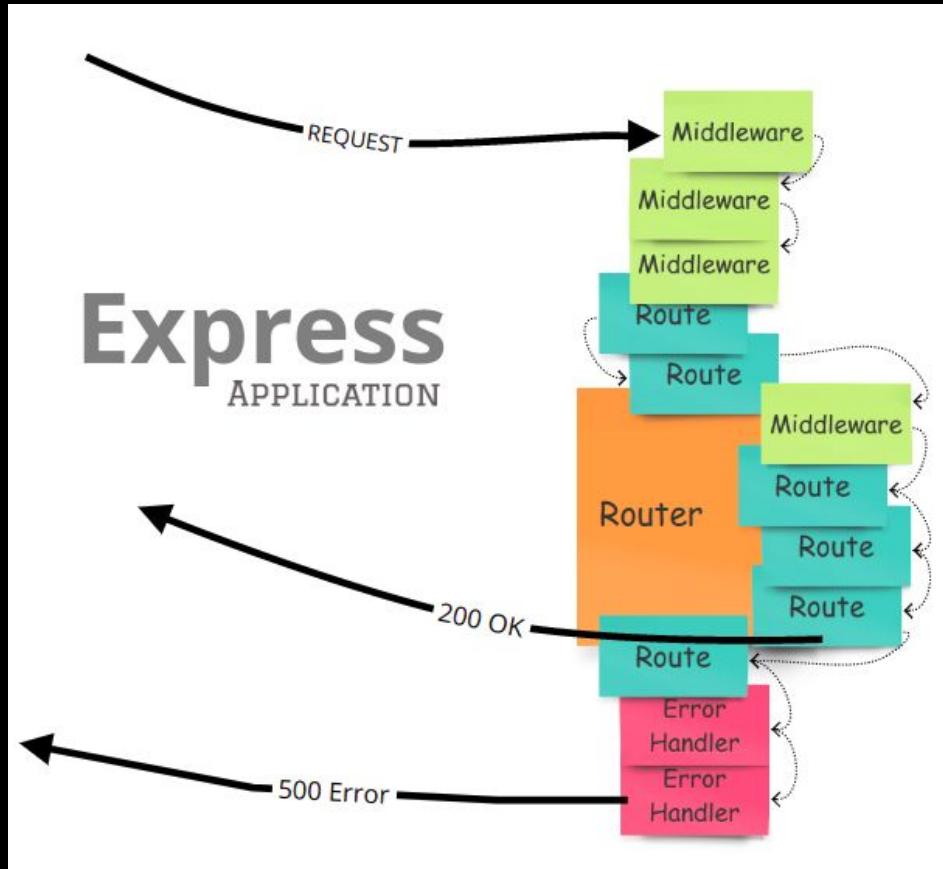
9. Starting with Express.js

1. What is Express.js
2. Need of Express.js
3. Installing Express.js
4. Adding Middleware
5. Sending Response
6. Express DeepDive
7. Handling Routes



9.1 What is Express.js

1. Express.js is a **minimal** and **flexible** web application framework for Node.js.
2. It provides a **robust** set of features for building single-page, multi-page, and hybrid web applications.
3. Express.js **simplifies server-side coding** by providing a layer of fundamental web application features.

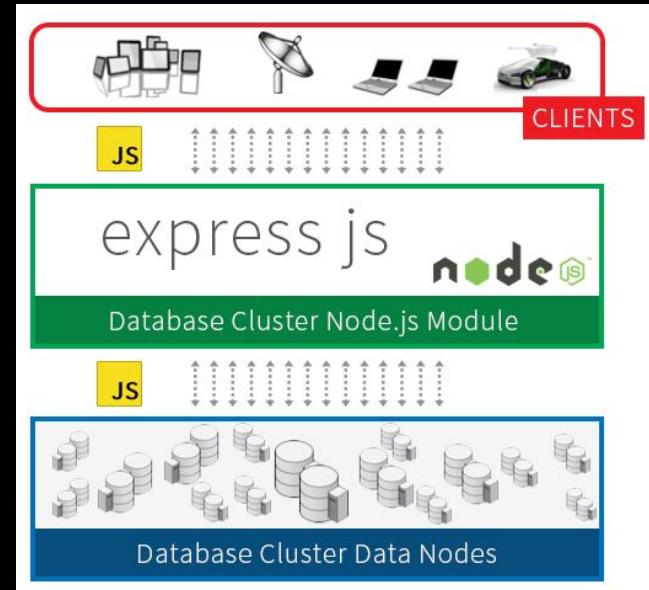




9.2 Need of Express.js

EXPRESS JS

1. **Express.js Simplifies Server Creation:** Helps in quickly setting up and running a web server without the need for complex coding.
2. **Routing Management:** Provides a powerful routing mechanism to handle URLs and HTTP methods effectively.
3. **Middleware Support:** Allows the use of middleware to handle requests, responses, and any middle operations, making code modular and maintainable.
4. **API Development:** Facilitates easy and efficient creation of RESTful APIs.
5. **Community and Plugins:** Has a large ecosystem with numerous plugins and extensions, accelerating development time.





9.3 Installing Express.js

EXPRESS **Js**

```
$ npm install express
```

To install Express temporarily and not add it to the dependencies list:

```
$ npm install express --no-save
```

By default with version npm 5.0+, `npm install` adds the module to the `dependencies` list in the `package.json` file; with earlier versions of npm, you must specify the `--save` option explicitly. Then, afterwards, running `npm install` in the app directory will automatically install modules in the dependencies list.



9.3 Installing Express.js

EXPRESS Js

```
// Core Modules
const http = require('http');
// External Modules
const express = require('express');

const app = express();

const server = http.createServer(app);

const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});
```

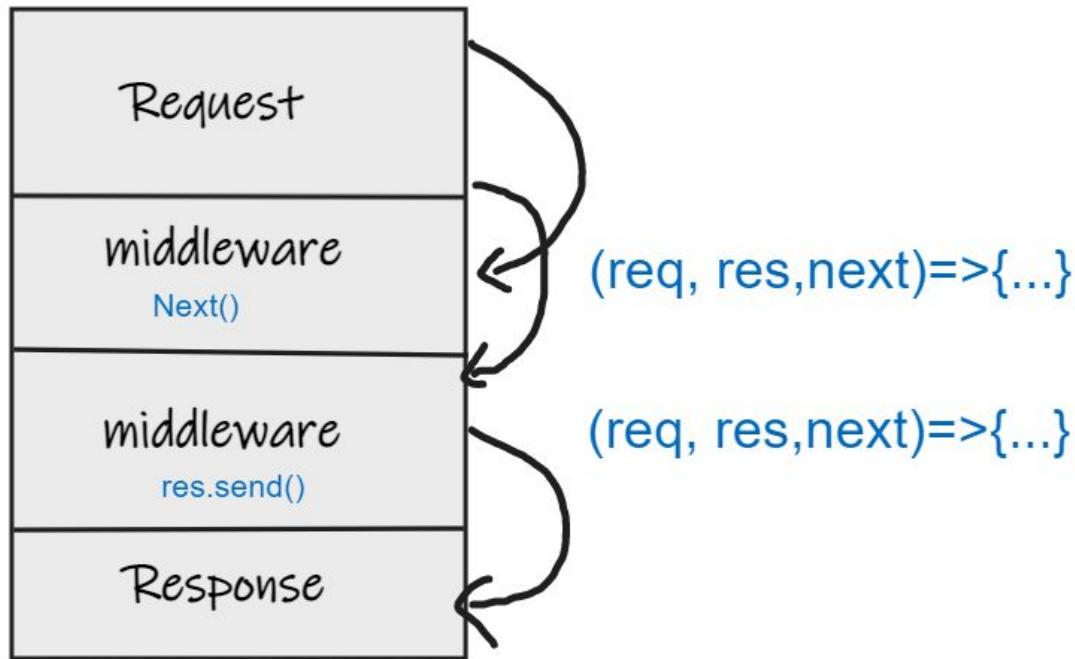
```
prashantjain@Prashants-Mac-mini node % npm start

> node@1.0.0 start
> nodemon app.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Server running at http://localhost:3000/
```

9.4 Adding Middleware

it is ALL About middleware





9.4 Adding Middleware

EXPRESS Js

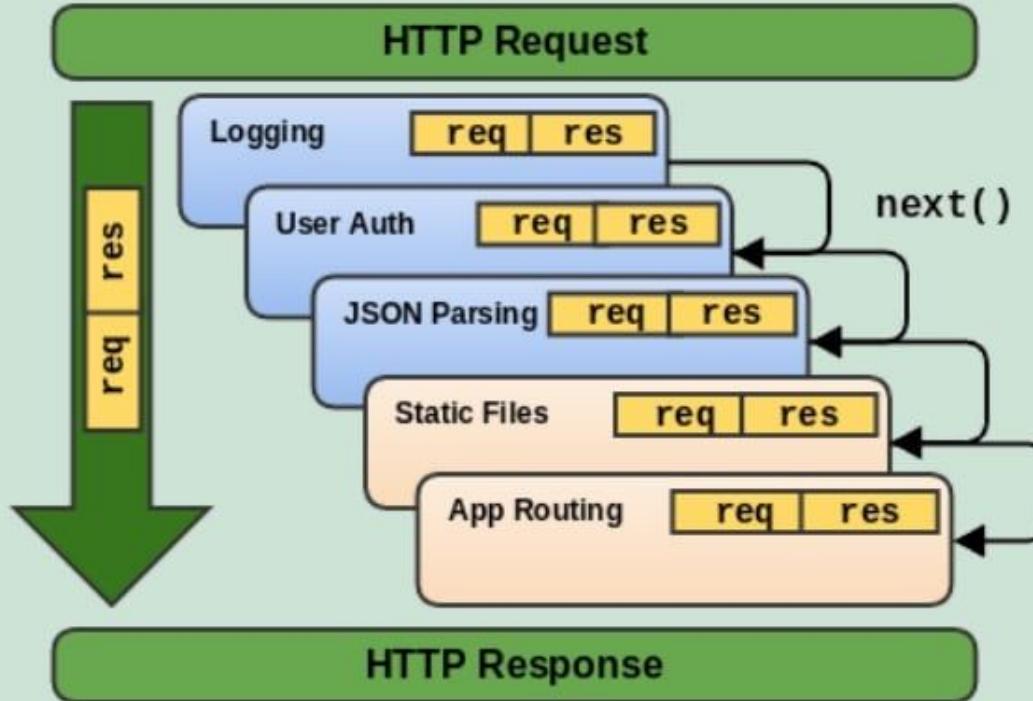
```
// Adding Middleware
app.use((req, res, next) => {
  console.log("First Middleware", req.url, req.method);
  next();
});

app.use((req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
});

const server = http.createServer(app);
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Server running at http://localhost:3000/
First Middleware / GET
Second Middleware / GET
```

9.4 Adding Middleware





9.5 Sending Response

EXPRESS Js

```
app.use((req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Complete Coding</p>');
});
```

A screenshot of a browser window. The address bar shows 'localhost:3000'. The main content area displays the text 'Welcome to Complete Coding'.

| ▼ Response Headers | |
|--------------------|------------------------------------|
| | <input type="checkbox"/> Raw |
| Connection: | keep-alive |
| Content-Length: | 33 |
| Content-Type: | text/html; charset=utf-8 |
| Date: | Wed, 02 Oct 2024 07:45:48 GMT |
| Etag: | W/"21-FoFd61RXqayGnfodqIQO62RgMzY" |
| Keep-Alive: | timeout=5 |
| X-Powered-By: | Express |



9.6 Express DeepDive

EXPRESS JS

expressjs / express

Type ⌘ to search

Code Issues 110 Pull requests 66 Discussions Actions Wiki Security 2 Insights

express Public Watch 1695

master 16 Branches 295 Tags Go to file Add file Code

| Author | Commit Message | Date |
|-------------------|--|--------------|
| jonchurch | remove --bail from test script (#5962) | yesterday |
| .github/workflows | update CI, remove unsupported versions, clean up | 3 weeks ago |
| benchmarks | docs: add documentation for benchmarks | 8 months ago |
| examples | Delete back as a magic string (#5933) | 3 weeks ago |
| lib | Delete back as a magic string (#5933) | 3 weeks ago |
| test | Delete back as a magic string (#5933) | 3 weeks ago |
| .editorconfig | build: Add .editorconfig | 7 years ago |



9.6 Express DeepDive

EXPRESS Js

```
101  /**
102   * Send a response.
103   *
104   * Examples:
105   *
106   *     res.send(Buffer.from('wahoo'));
107   *     res.send({ some: 'json' });
108   *     res.send('<p>some html</p>');
109   *
110  * @param {string|number|boolean|object|Buffer} body
111  * @public
112  */
113
114  res.send = function send(body) {
115    var chunk = body;
116    var encoding;
117    var req = this.req;
118    var type;
119
120    // settings
121    var app = this.app;
122
```

```
// string defaulting to html
case 'string':
  if (!this.get('Content-Type')) {
    this.type('html');
  }
break;
case 'boolean':
```



9.6 Express DeepDive

EXPRESS Js

```
// Core Modules
const http = require('http');
// External Modules
const express = require('express');

const app = express();

// Adding Middleware
app.use((req, res, next) => {
  console.log("First Middleware", req.url, req.method);
  next();
});

app.use((req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Complete Coding</p>');
});

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});
```



9.7 Handling Routes

EXPRESS Js

app.use([path,] callback [, callback...])

Mounts the specified [middleware](#) function or functions at the specified path: the middleware function is executed when the base of the requested path matches `path`.

Arguments

| Argument | Description | Default |
|-------------------|---|-----------------|
| <code>path</code> | <p>The path for which the middleware function is invoked; can be any of:</p> <ul style="list-style-type: none">• A string representing a path.• A path pattern.• A regular expression pattern to match paths.• An array of combinations of any of the above. <p>For examples, see Path examples.</p> | '/' (root path) |



9.7 Handling Routes

EXPRESS Js

```
const app = express();
app.use('/', (req, res, next) => {
  console.log("Hello World");
  //res.send('<p>Welcome to Submit Details Page</p>');
  next();
});

app.use('/submit-details', (req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Submit Details Page</p>');
});

app.use('/', (req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Complete Coding</p>');
});
```

1. Order matters
2. Can not call `next()` after `send()`
3. "/" matches everything
4. Calling `res.send` implicitly calls `res.end()`





React

9.7 Handling Routes

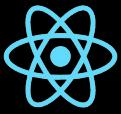
EXPRESS Js

app.use([path,] callback [, callback...])

Mounts the specified [middleware](#) function or functions at the specified path: the middleware function is executed when the base of the requested path matches `path`.

Arguments

| Argument | Description | Default |
|-------------------|---|-----------------|
| <code>path</code> | <p>The path for which the middleware function is invoked; can be any of:</p> <ul style="list-style-type: none">• A string representing a path.• A path pattern.• A regular expression pattern to match paths.• An array of combinations of any of the above. <p>For examples, see Path examples.</p> | '/' (root path) |



React

```
const app = express();
app.use('/', (req, res, next) => {
  console.log("Hello World");
  //res.send('<p>Welcome to Submit Details Page</p>');
  next();
});

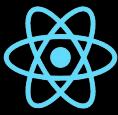
app.use('/submit-details', (req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Submit Details Page</p>');
});

app.use('/', (req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Complete Coding</p>');
});
```

EXPRESS Js

1. Order matters
2. Can not call `next()` after `send()`
3. "/" matches everything
4. Calling `res.send` implicitly calls `res.end()`





React

9.7 Handling Routes

EXPRESS Js

```
var express = require('express');
var app = express();
app.get('/', function(req, res, next) {
  next();
})
app.listen(3000);
```

HTTP method for which the middleware function applies.

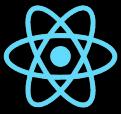
Path (route) for which the middleware function applies.

The middleware function.

Callback argument to the middleware function, called "next" by convention.

HTTP response argument to the middleware function, called "res" by convention.

HTTP request argument to the middleware function, called "req" by convention.



React

9.7 Handling Routes

EXPRESS Js

```
const app = express();
app.post('/', (req, res, next) => {
    console.log("Hello World");
    //res.send('<p>Welcome to Submit Details Page</p>');
    next();
});

app.use('/submit-details', (req, res, next) => {
    console.log("Second Middleware", req.url, req.method);
    res.send('<p>Welcome to Submit Details Page</p>');
});

app.use('/', (req, res, next) => {
    console.log("Second Middleware", req.url, req.method);
    res.send('<p>Welcome to Complete Coding</p>');
});
```



Practise Set

Create a new project.

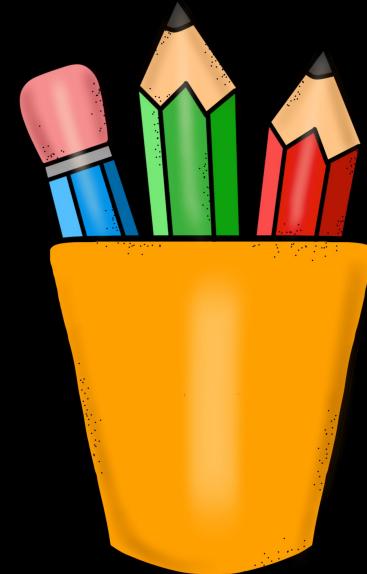
1. Install nodemon and express.
2. Add two dummy middleware that logs request path and request method respectively.
3. Add a third middleware that returns a response.
4. Now add handling using two more middleware that handle path /, a request to /contact-us page.
5. Contact us should return a form with name and email as input fields that submits to /contact-us page also.
6. Also handle POST incoming request to /contact-us path using a separate middleware.

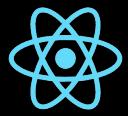




Revision

1. What is Express.js
2. Need of Express.js
3. Installing Express.js
4. Adding Middleware
5. Sending Response
6. Express DeepDive
7. Handling Routes





React

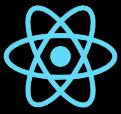


React

10. Express.js DeepDive

1. Parsing Requests
2. Express Router
3. Adding 404
4. Common Paths
5. Adding HTML Files
6. Serving HTML Files
7. Using File Helper





React

// External Modules

```
const express = require('express');
const bodyParser = require('body-parser');
```

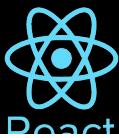
```
const app = express();
```

```
app.use(bodyParser.urlencoded());
```

```
app.get('/details', (req, res, next) => {
  console.log("Hello World");
  res.send(` 
    <h1>Enter Your Details:</h1>
    <form action="/submit-details" method="POST">
      <input type="text" name="username" placeholder="Enter your name"><br>
      <br><input type="submit" value="Submit">
    </form>
  `);
});
```

```
app.post('/submit-details', (req, res, next) => {
  console.log(req.url, req.method, req.body);
  res.redirect('/');
});
```

EXPRESS Js



EXPRESS Js



Stays

Experiences

Airbnb your home



Where

Search destinations

Check in

Add dates

Check out

Add dates

Who

Add guests



Icons



Amazing pools



Farms



Amazing views



Surfing



Ski-in/out



Rooms



Cabins



Countryside



Treehouses



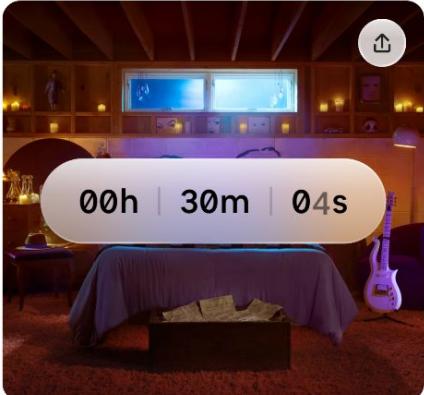
OMG!



Luxe



Beach



Stay in Prince's Purple Rain house
Hosted by Wendy And Lisa
₹564 per guest



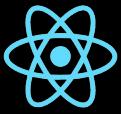
Join the Living Room Session with Do...
Hosted by Doja Cat
Coming 8 October



Sleepover at Polly Pocket's Compact
Hosted by Polly Pocket
Sold out



Playdate at Polly Pocket's Compact
Hosted by Polly Pocket
Sold out

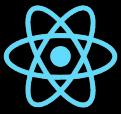


React

node > air-bnb > app.js > ...

```
1 // External Modules
2 const express = require('express');
3 // Local Modules
4 const hostRouter = require('./routes/host');
5 const userRouter = require('./routes/user');
6
7 const app = express();
8
9 app.use(express.urlencoded({extended:true}));
10 app.use(hostRouter);
11 app.use(userRouter);
12
13 const PORT = 3000;
14 app.listen(PORT, () => {
15   | console.log(`Server running at http://localhost:${PORT}/` );
16 });
```

EXPRESS

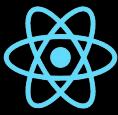


React

node > air-bnb > app.js > ...

```
1 // External Modules
2 const express = require('express');
3 // Local Modules
4 const hostRouter = require('./routes/host');
5 const userRouter = require('./routes/user');
6
7 const app = express();
8
9 app.use(express.urlencoded({extended:true}));
10 app.use(hostRouter);
11 app.use(userRouter);
12
13 const PORT = 3000;
14 app.listen(PORT, () => {
15   | console.log(`Server running at http://localhost:${PORT}/` );
16 });
```

EXPRESS



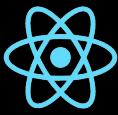
React

10.2 Express Router

EXPRESS Js

node > air-bnb > routes > host.js > ...

```
1 // External Modules
2 const express = require('express');
3
4 const router = express.Router();
5
6 router.get('/add-home', (req, res, next) => {
7   res.send(`
8     <h1>Register Your Home on AirBnB</h1>
9     <form action="/submit-home" method="POST">
10       <input type="text" name="houseName" placeholder="Enter your House Name"><br>
11       <br><input type="submit" value="Submit">
12     </form>
13   `);
14 });
15
16 router.post('/submit-home', (req, res, next) => {
17   console.log("Post Request details", req.url, req.method, req.body);
18   res.redirect('/');
19 });
20
21 module.exports = router;
```



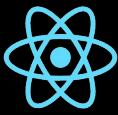
React

10.2 Express Router

EXPRESS Js

node > air-bnb > routes > `js` user.js > ...

```
1 // External Modules
2 const express = require('express');
3 const userRouter = express.Router();
4
5 userRouter.use('/', (req, res, next) => {
6   console.log("Second Middleware", req.url, req.method);
7   res.send(`
8     <h2>Welcome to AirBnb</h2>
9     <a href="/add-home">Register Your Home on AirBnb</a>
10    `);
11 });
12
13 module.exports = userRouter;
```

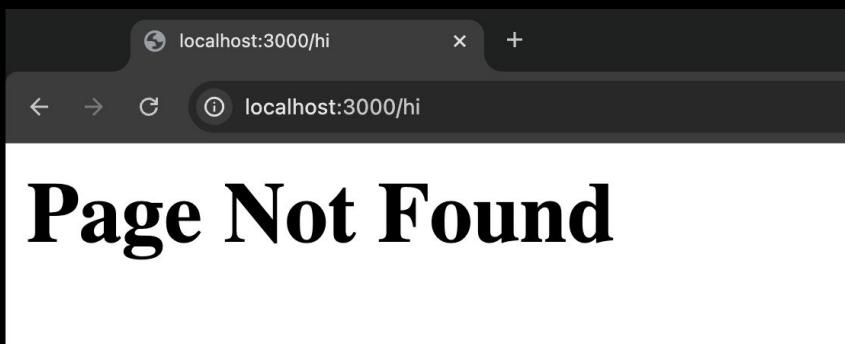


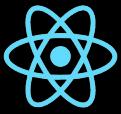
React

10.3 Adding 404

EXPRESS Js

```
app.use(express.urlencoded({extended:true}));  
  
app.use(hostRouter);  
app.use(userRouter);  
  
app.use((req, res, next) => {  
  res.status(404).send('<h1>Page Not Found</h1>');  
});  
  
const PORT = 3000;  
app.listen(PORT, () => {  
  console.log(`Server running at http://localhost:\${PORT}/`);  
});
```





React

10.4 Common Paths

EXPRESS Js

```
// GET path
router.get('/host/add-home', (req, res, next) => {
  res.send(`<h1>Register Your Home on AirBnB</h1>
<form action="/add-home" method="POST">
  <input type="text" name="houseName"
    placeholder="Enter your House Name"><br>
  <br><input type="submit" value="Submit">
</form>
`);
});
```

```
// POST path
router.post('/host/add-home', (req, res, next) => {
  console.log("Post Request details", req.url, req.
  method, req.body);
  res.redirect('/');
});
```

or

```
const app = express();

app.use(express.urlencoded({extended:true}));

app.use("/host", hostRouter);
app.use("/user", userRouter);

app.use((req, res, next) => {
  res.status(404).send('<h1>Page Not Found</h1>');
});
```



React

10.5 Adding HTML Files

EXPRESS Js

air-bnb > views > add-home.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0" />
  <title>airbnb</title>
</head>
<body>
  <header>
    <nav>
      <ul>
        <li><a href="/">Go to Home</a></li>
        <li><a href="/host/add-home">Add Home</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <h1>Register Your Home on AirBnB</h1>
    <form action="/host/add-home" method="POST">
      <input
        type="text"
        name="houseName"
        placeholder="Enter your House Name"
      /><br />
      <br /><input type="submit" value="Submit" />
    </form>
  </main>
</body>
</html>
```

air-bnb > views > user.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0" />
  <title>airbnb</title>
</head>
<body>
  <header>
    <nav>
      <ul>
        <li><a href="/">Go to Home</a></li>
        <li><a href="/host/add-home">Add Home</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <h2>Welcome to AirBnb</h2>
  </main>
</body>
</html>
```

air-bnb

> node_modules

> routes

host.js

user.js

> views

404.html

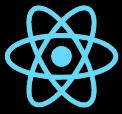
add-home.html

user.html

app.js

package-lock.json

package.json



React

10.6 Serving HTML Files

EXPRESS Js

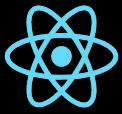
air-bnb > routes > `host.js` > ...

```
// Core Modules
const path = require('path');
```

```
// External Modules
const express = require('express');
```

```
const router = express.Router();
```

```
// GET path
router.get('/add-home', (req, res, next) => {
  res.sendFile(path.join(__dirname, '../', 'views', 'add-home.html'));
});
```



React

airbnb > util > `path.js` > ...

```
// Core Modules
const path = require('path');
```

```
module.exports = path.dirname(require.main.filename);
```

```
// Local Modules
```

```
const rootDir = require('../util/path');
```

```
userRouter.get('/', (req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.sendFile(path.join(rootDir, 'views', 'user.html'));
});
```

EXPRESS Js

A file tree visualization for the 'airbnb' project. The root folder contains 'node_modules', 'routes', 'util', and 'views'. 'util' contains 'host.js' and 'user.js'. 'views' contains 'app.js'. Additionally, there are 'package-lock.json' and 'package.json' files at the root level.

```
airbnb
├── node_modules
└── routes
    ├── host.js
    └── user.js
└── util
    ├── host.js
    └── user.js
└── views
    ├── app.js
    └── package-lock.json
    └── package.json
```



React

Practise Set

Reuse the app from the last assignment

1. Parse the body of the contact-us request and log it to console.
2. Move the code to separate local modules and use the Express router to import and use them in app.js
3. Move all the html code to html files and serve them using the file helper.
4. Also add a 404 page for this app.

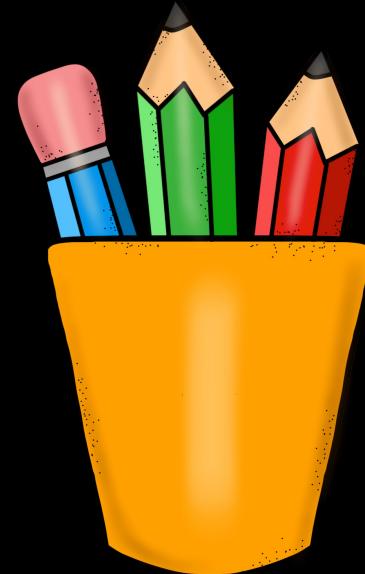


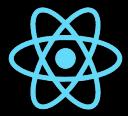


React

Revision

1. Parsing Requests
2. Express Router
3. Adding 404
4. Common Paths
5. Adding HTML Files
6. Serving HTML Files
7. Using File Helper



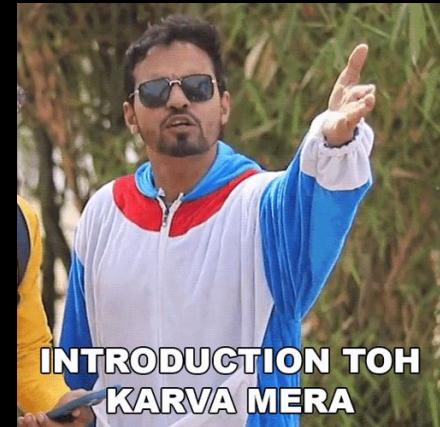


React

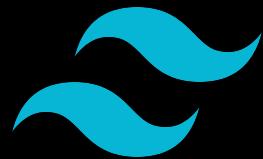


11. Styling using tailwindcss

1. Serving Static Files
2. Introduction to Tailwind CSS
3. Utility Classes
4. Installing Extension
5. Including Tailwind CSS
6. Installing Tailwind CSS
7. Using Tailwind CSS
8. Building Tailwind Automatically



11.1 Serving Static Files



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>airbnb</title>
    <link rel="stylesheet" href="airbnb.css">
  </head>
  <body>
    <header>
      <nav>
        <ul>
          <li><a href="/">Go to Home</a></li>
          <li><a href="/host/add-home">Add Home</a></li>
        </ul>
      </nav>
    </header>
    <main>
      <h2>Welcome to AirBnb</h2>
    </main>
  </body>
</html>
```

```
body {
  font-family: 'Arial', sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f5f5f5;
}

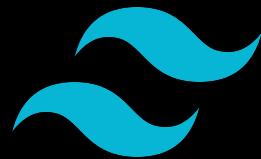
header {
  background-color: #ff5a5f;
  padding: 1rem 2rem;
  box-shadow: 0px 2px 8px rgba(0, 0, 0, 0.1);
}

nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
}

nav ul {
  list-style: none;
  margin: 0;
  padding: 0;
  display: flex;
}

nav li {
  margin-left: 20px;
}
```

11.1 Serving Static Files



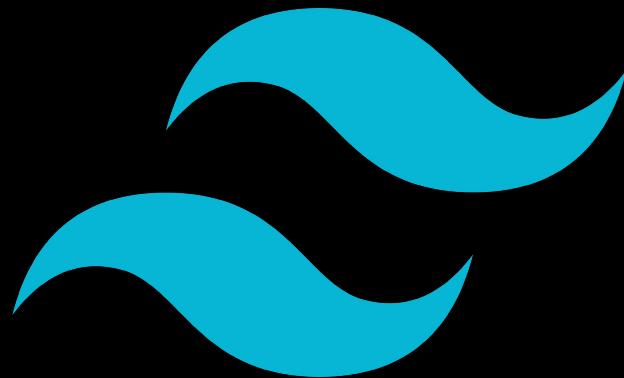
```
nav a {  
    text-decoration: none;  
    color: black;  
    font-weight: bold;  
    padding: 0.5rem 1rem;  
    border-radius: 4px;  
    transition: background-color 0.3s ease;  
}  
  
nav a:hover {  
    background-color: #e54e52;  
}  
  
main {  
    text-align: center;  
    padding: 5rem 2rem;  
    background-color: white;  
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
    margin: 2rem auto;  
    max-width: 600px;  
    border-radius: 8px;  
}  
  
h2 {  
    color: black;  
    font-size: 2rem;  
}
```



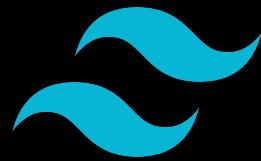
```
// Granting access to public folder  
app.use(express.static(path.join(rootDir, "public")));
```

11.2 Introduction to Tailwind CSS

1. **Responsive:** Mobile-first design for all device sizes.
2. **Utility-First:** Provides low-level utility classes for building custom designs.
3. **Highly Customizable:** Easily extendable through a config file.
4. **Responsive Design:** Built-in responsive utilities (e.g., sm:, md:).
5. **No Predefined Components:** Focuses on building custom components.
6. **Purge CSS:** Removes unused styles in production for smaller files.
7. **Fast Development:** Style elements directly in markup for speed.



11.3 Utility Classes



```
1 /* Colors */
2 .text-primary { color: #007bff; }
3 .bg-primary { background-color: #007bff; }
4
5 /* Sizing */
6 .w-full { width: 100%; }
7 .h-full { height: 100%; }
8
9 /* Typography */
10 .text-center { text-align: center; }
11 .font-bold { font-weight: bold; }
12
13 /* Spacing */
14 .m-1 { margin: 0.25rem; }
15 .m-2 { margin: 0.5rem; }
16 .p-1 { padding: 0.25rem; }
17 .p-2 { padding: 0.5rem; }
```

```
19 /* Layout */
20 .d-flex { display: flex; }
21 .flex-col { flex-direction: column; }
22 .items-center { align-items: center; }
23 .justify-center { justify-content: center; }
24
25 /* Misc */
26 .rounded { border-radius: 0.25rem; }
27 .hidden { display: none; }
```

11.4 Installing Extension



 tailwindcss

INTELLISENSE

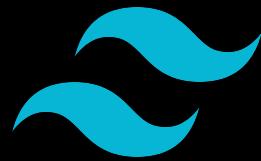
FOR



Visual Studio Code

```
class="bg-t| md:flex md:items-center md:  
v class="fl □ bg-teal-50  
<h2 class="t □ bg-teal-100  
    Back End D □ bg-teal-200  
</h2> □ bg-teal-300      background  
</div> □ bg-teal-400  
div class="mt □ bg-teal-500  
<span class= □ bg-teal-600  
    <button ty □ bg-teal-700  
        Edit □ bg-teal-800  
    </button> □ bg-teal-900  
</span> □ bg-transparent  
<span class= □ bg-top  
    <button type="button" class="in □ bg-line-t  
        Publish □  
    </button>  
</span>
```

11.4 Installing Extension



Tailwind CSS IntelliSense v0.12.10

Tailwind Labs [tailwindcss.com](#) | ⚡ 7,381,739 | ★★★★★ (99)

Intelligent Tailwind CSS tooling for VS Code

Install



Auto Update



DETAILS

FEATURES

CHANGELOG

A screenshot of the Visual Studio Code interface. The code editor shows a snippet of Tailwind CSS code with several class names highlighted. A dropdown menu is open over one of the highlighted classes, showing suggestions like "bg-teal-50", "bg-teal-100", "bg-teal-200", "bg-teal-300", "bg-teal-400", "bg-teal-500", "bg-teal-600", "bg-teal-700", "bg-teal-800", and "bg-teal-900". The background of the code editor has a grid pattern. At the bottom, there are two logos: the Tailwind CSS logo and the Visual Studio Code logo.

Categories

Linters

Resources

Marketplace

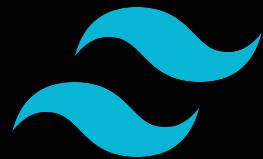
Issues

Repository

License

Tailwind Labs

11.5 Including Tailwind CSS



Installation

Get started with Tailwind CSS

Tailwind CSS works by scanning all of your HTML files, JavaScript components, and any other templates for class names, generating the corresponding styles and then writing them to a static CSS file.

It's fast, flexible, and reliable — with zero-runtime.

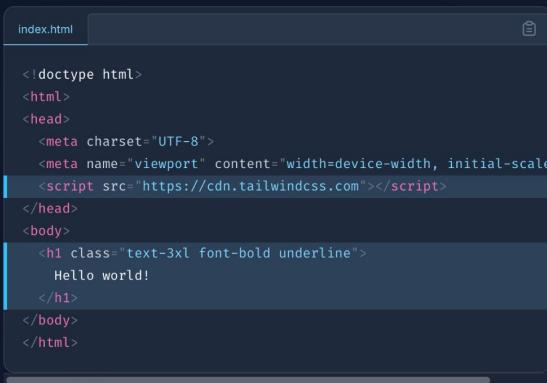
Installation

[Tailwind CLI](#) [Using PostCSS](#) [Framework Guides](#) [Play CDN](#)

Use the Play CDN to try Tailwind right in the browser without any build step. The Play CDN is designed for development purposes only, and is not the best choice for production.

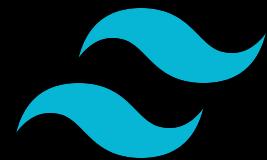
1 Add the Play CDN script to your HTML

Add the Play CDN script tag to the `<head>` of your HTML file, and start using Tailwind's utility classes to style your content.



```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
  <h1 class="text-3xl font-bold underline">
    Hello world!
  </h1>
</body>
</html>
```

11.6 Installing Tailwind CSS



1. Install:

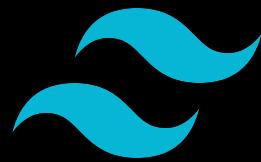
```
npm init -y
```

```
npm install -D tailwindcss postcss autoprefixer
```

2. Initialize Tailwind CSS Config

```
npx tailwindcss init
```

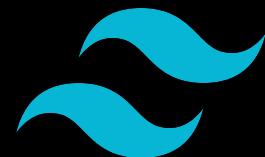
11.6 Installing Tailwind CSS



Configure Tailwind in the Configuration Files (tailwind.config.js)

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  ...
  content: ["./views/*.html"],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

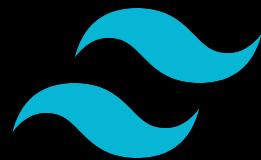
11.6 Installing Tailwind CSS



Add Tailwind Directives to views/input.css

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

11.6 Installing Tailwind CSS



5. Include public/output.css into your index.html

6. Run Command

```
npx tailwindcss -i ./views/input.css -o ./public/output.css --watch
```

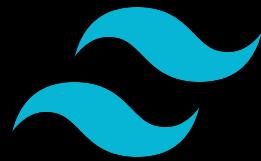
7. Declare Shortcut:

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "tailwind": "npx tailwindcss -i ./src/input.css -o ./src/output.css --watch"  
},
```

8. Run Command

```
npm run tailwind
```

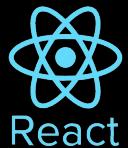
11.7 Using Tailwind CSS



```
<body class="■bg-gray-100 font-sans">
<header class="■bg-red-500 ■text-white p-4 shadow-md">
  <nav class="container mx-auto flex justify-between items-center">
    <ul class="flex space-x-4">
      <li>
        <a href="/" class="■hover:bg-red-400 py-2 px-4 rounded transition duration-300">Go to Home</a>
      </li>
      <li>
        <a href="/host/add-home" class="■hover:bg-red-400 py-2 px-4 rounded transition duration-300">Add Home</a>
      </li>
    </ul>
  </nav>
</header>
<main class="container mx-auto ■bg-white shadow-lg rounded-lg p-8 mt-10 max-w-xl">
  <h2 class="text-2xl ■text-gray-800 font-bold text-center mb-6">
    Welcome to Airbnb
  </h2>
</main>
</body>
```

11.8 Building Tailwind Automatically

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "tailwind": "tailwindcss -i ./views/input.css -o ./public/output.css --watch",  
  "start": "nodemon app.js & npm run tailwind"  
},
```



Practise Set

Style other page using
Tailwind.

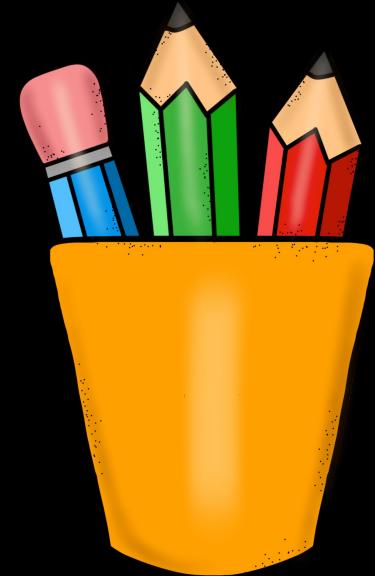


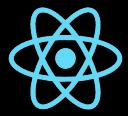


React

Revision

1. Serving Static Files
2. Introduction to Tailwind CSS
3. Utility Classes
4. Installing Extension
5. Including Tailwind CSS
6. Installing Tailwind CSS
7. Using Tailwind CSS





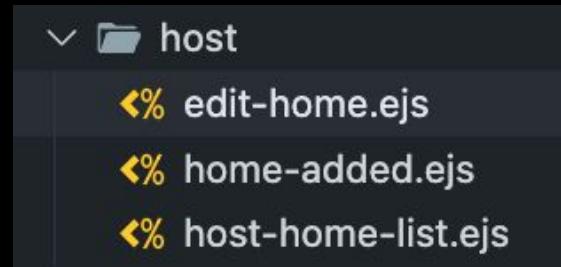
React

14.6 Edit-Home: Adding Feature Skeleton

1. Rename the `add-home.ejs` to `edit-home.ejs`
2. Fix it's usage in `the add-home controller`.
3. Change the `path of edit button` everywhere.
4. Add a new router for `GET /edit-home/:home-id`
5. Add a `new controller method` in host controller, optionally logging query param of `editing=true`, while passing editing flag to view.

14.6 Edit-Home: Adding Feature Skeleton

1.



2.

```
exports.getAddHome = (req, res, next) => {
  res.render("host/edit-home", {
    pageTitle: "Add Home to airbnb",
    currentPage: "addHome",
  });
};
```

14.7 Edit-Home: Showing Existing Data

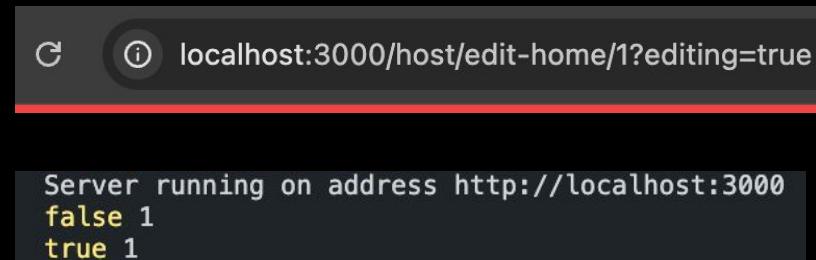
3.

```
<%- include('../partials/head') %>
<body>
  <%- include('../partials/nav') %>
  <main class="container mx-auto bg-white shadow-lg rounded-lg p-8 mt-10 max-w-6xl">
    <div class="flex flex-wrap justify-center gap-6">
      </div>
      <a href="/host/edit-home/<%= home.id %>?editing=true" class="bg-blue-500 p-2">Edit</a>
      <button class="bg-red-500 p-2">Delete</button>
    </div>
    </div>
  <% }) %>
</div>
</main>
</body>
</html>
```

14.6 Edit-Home: Adding Feature Skeleton

4. hostRouter.get("/edit-home/:homeId", hostController.getEditHome);

```
5. exports.getEditHome = (req, res, next) => {
    const editing = req.query.editing === "true";
    const homeId = req.params.homeId;
    console.log(editing, homeId);
    res.render("host/edit-home", {
        pageTitle: "Edit Home",
        currentPage: "editHome",
        editing: editing,
        homeId: homeId,
    });
};
```



A screenshot of a terminal window showing Node.js server logs. The title bar says "localhost:3000/host/edit-home/1?editing=true". The log output shows:

```
C i localhost:3000/host/edit-home/1?editing=true
Server running on address http://localhost:3000
false 1
true 1
```

14.7 Edit-Home: Showing Existing Data

1. Change the controller to now fetch the home details, if not found redirect to /host/host-home-list otherwise passing the data to view.
2. Change the edit-home.ejs to show dynamic content based on values:
 - a. Different submit path.
 - b. Different button text.
 - c. Pre-filled values if present.

14.7 Edit-Home: Showing Existing Data

```
1. exports.getEditHome = (req, res, next) => {
    const editing = req.query.editing === "true";
    const homeId = req.params.homeId;
    Home.findById(homeId, (home) => {
        if (!home) {
            return res.redirect("/host/host-home-list");
        }
        res.render("host/edit-home", {
            pageTitle: "Edit Home",
            currentPage: "editHome",
            editing: editing,
            home: home,
        });
    });
};
```

14.7 Edit-Home: Showing Existing Data

2.

```
<%- include('../partials/head') %>
</head>
<body>
  <%- include('../partials/nav') %>
  <main class="container mx-auto mt-8 p-8 bg-white rounded-lg shadow-md">
    <h1 class="text-3xl font-bold mb-6 text-center text-gray-800">Register Your Home on AirBnB</h1>
    <form action="/host/<%= editing ? 'edit-home' : 'add-home' %>" method="POST" class="max-w-md mx-auto">
      <input
        type="text"
        name="houseName"
        value="<%= home ? home.houseName : '' %>"
        placeholder="Enter your House Name"
        class="w-full px-4 py-2 mb-4 border rounded-md focus:outline-none focus:ring-2 focus:ring-red-500"/>
      <input
        type="text"
        name="price"
        value="<%= home ? home.price : '' %>"
        placeholder="Price Per Night"
        class="w-full px-4 py-2 mb-4 border rounded-md focus:outline-none focus:ring-2 focus:ring-red-500"/>
      <input
        type="text"
        name="location"
        value="<%= home ? home.location : '' %>"
        placeholder="Location"
        class="w-full px-4 py-2 mb-4 border rounded-md focus:outline-none focus:ring-2 focus:ring-red-500"/>
      <input
        type="text"
        name="rating"
        value="<%= home ? home.rating : '' %>"
        placeholder="Rating"
        class="w-full px-4 py-2 mb-4 border rounded-md focus:outline-none focus:ring-2 focus:ring-red-500"/>
      <input
        type="text"
        name="photoUrl"
        value="<%= home ? home.photoUrl : '' %>"
        placeholder="Enter Photo URL"
        class="w-full px-4 py-2 mb-4 border rounded-md focus:outline-none focus:ring-2 focus:ring-red-500"/>
      <input type="submit" value="<%= editing ? 'Update' : 'Add' %> Home" class="w-full bg-red-500 text-white py-2 transition duration-300"/>
    </form>
  </main>
</body>
</html>
```

14.8 Edit-Home: Handing Edit Request

1. Add a router for POST /edit-home
2. Add a controller for /edit-home, which creates a home model object and saves it, before redirecting to /host/host-home-list
3. Add hidden id input field in the edit-home.ejs to get the id as part of POST request.
4. Change the save method in Model to handle creation of new as well as updation of existing Home.

14.8 Edit-Home: Handing Edit Request

1. hostRouter.post("/edit-home", hostController.postEditHome);

```
exports.postEditHome = (req, res, next) => {
  const { id, houseName, price, location, rating, photoUrl } = req.body;
  const home = new Home(houseName, price, location, rating, photoUrl);
  home.id = id;
  home.save();
  res.redirect("/host/host-home-list");
};
```

3.

```
<%- include('../partials/head') %>
<body>
  <%- include('../partials/nav') %>
  <main class="container mx-auto mt-8 p-8 bg-white rounded-lg shadow-md">
    <h1 class="text-3xl font-bold mb-6 text-center text-gray-800">Register
    <form action="/host/<%= editing ? 'edit-home' : 'add-home' %>" method="P
      <input type="hidden" name="id" value="<% home ? home.id : '' %>">
      <input
        type="text"
        name="houseName"
```

14.8 Edit-Home: Handing Edit Request

4.

```
module.exports = class Home {  
  save() {  
    Home.fetchAll((registeredHomes) => {  
      console.log(registeredHomes, this);  
      if (this.id) {  
        registeredHomes = registeredHomes.map((home) => {  
          console.log(home.id, this.id);  
          if (home.id === this.id) {  
            return this;  
          }  
          return home;  
        });  
      } else {  
        console.log("New Home");  
        this.id = Math.random().toString();  
        registeredHomes.push(this);  
      }  
      const homeDataPath = path.join(rootDir, "data", "homes.json");  
      fs.writeFile(homeDataPath, JSON.stringify(registeredHomes), (error) => {  
        console.log("File Writing Concluded", error);  
      });  
    });  
  }  
}
```

14.9 Adding Delete Feature

1. Surround the **delete button** with a **form** that submits to path
`/host/delete-home/:homeld`
2. Add a **route** in the host routes.
3. Add a **static delete** method to the **Home model** that takes an id and deletes the home.
4. Add a method in **host controller** to handle the request, **delete the home** and redirect to `/host/host-homes-list` page.
5. Pending: **deleted homes** might still be in **favourites list**.

14.9 Adding Delete Feature

1.

```
<a href="/host/edit-home/<%= home.id %>?editing=true" class="■bg-blue-500 p-2 rounded
■text-white ■hover:bg-blue-600">Edit</a>
<form action="/host/delete-home/<%= home.id %>" method="POST" class="inline">
  <button type="submit" class="■bg-red-500 p-2 rounded ■text-white
■hover:bg-red-600">Delete</button>
</form>
```
2.

```
hostRouter.post("/delete-home/:homeId", hostController.postDeleteHome);
```
3.

```
static deleteById(id, callback) {
  this.fetchAll((homes) => {
    const updatedHomes = homes.filter((home) => home.id !== id);
    fs.writeFile(path.join(rootDir, "data", "homes.json"), JSON.stringify(updatedHomes), callback);
  });
}
```

14.9 Adding Delete Feature

```
4. exports.postDeleteHome = (req, res, next) => {
  const homeId = req.params.homeId;
  Home.deleteById(homeId, (error) => {
    if (error) {
      console.log("Error Deleting Home", error);
    }
    res.redirect("/host/host-home-list");
  });
};
```

14.10 Removing Home from Favourites

1. Add a static method to the favourites model to delete the home.
2. Add a form to the favourites-list in each home to path
`/favourites/delete/:homeld`
3. Add a route for POST delete-favourites in the store routes.
4. Add a method in store controller to use the model's static method and then redirect to favourites-list.
5. Use this model method in home-delete to make sure any deleted home is also removed from favourites.

14.10 Removing Home from Favourites

1.

```
static deleteFavourite(homeId, callback) {
  this.getFavourites((favourites) => {
    const updatedFavourites = favourites.filter((id) => id !== homeId);
    fs.writeFile(favouritesPath, JSON.stringify(updatedFavourites), callback);
  });
}
```
2.

```
<form action="/favourites/delete/<%= home.id %>" method="POST">
  <button type="submit" class="■bg-red-500 p-2 rounded ■text-white
  ■hover:■bg-red-600">Delete</button>
</form>
```
3.

```
storeRouter.post("/favourites/delete/:homeId", storeController.deleteFavourite);
```

14.10 Removing Home from Favourites

4.

```
exports.deleteFavourite = (req, res, next) => {
  const homeId = req.params.homeId;
  Favourites.deleteFavourite(homeId, (error) => {
    if (error) {
      console.log("Error Deleting Favourite", error);
    }
    res.redirect('/favourites');
  });
};
```

5.

```
static deleteById(id, callback) {
  this.fetchAll((homes) => {
    const updatedHomes = homes.filter((home) => home.id !== id);
    fs.writeFile(path.join(rootDir, "data", "homes.json"), JSON.stringify(updatedHomes), (error) => {
      Favourites.deleteFavourite(id, callback);
    });
  });
}
```



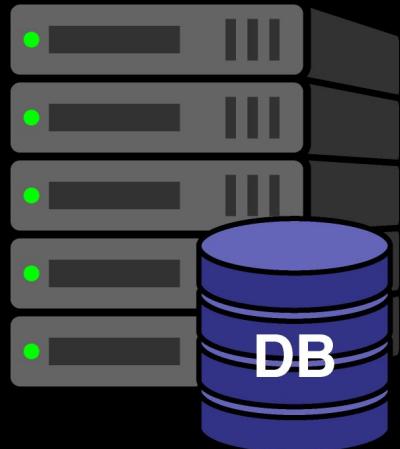
15. Introduction to SQL

1. What is a DB (Database)
2. Introduction to SQL DB
3. Introduction to NoSQL DB
4. SQL vs NoSQL
5. Installing MySQL
6. Connecting App to DB
7. Creating homes Table
8. Querying homes in App
9. Adding DB in Models
10. Adding Home in Model
11. Implementing Model using Where

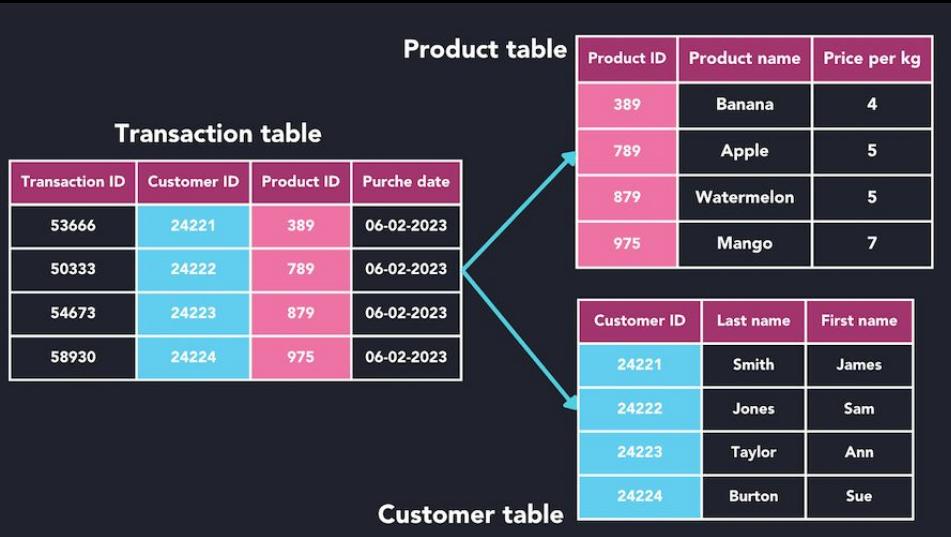


15.1 What is a DB (Database)

1. Store Data: Keep large amounts of data in a structured format.
2. Enable Data Management: Allow for adding, updating, and deleting data easily.
3. Facilitate Quick Access: Provide fast retrieval of data through queries.
4. Ensure Data Integrity: Maintain accuracy and consistency of data over time.
5. Support Multiple Users: Handle concurrent access by many users simultaneously.
6. Secure Data: Protect information through access controls and authentication.



15.2 Introduction to SQL DB



- **Vertical Scalability:** Typically scaled by increasing the resources of a single server (scaling up).
- **Relationships:** Tables can have multiple types of relationships.

- **Relational Model:** Organize data into tables with rows and columns.
- **Fixed Schema:** Require a predefined schema; the structure of the data must be known in advance.

15.2 Introduction to SQL DB MySQL™

Table: Customers

| customer_id | first_name | last_name | age | country |
|-------------|------------|-----------|-----|---------|
| 1 | John | Doe | 31 | USA |
| 2 | Robert | Luna | 22 | USA |
| 3 | David | Robinson | 22 | UK |
| 4 | John | Reinhardt | 25 | UK |
| 5 | Betty | Doe | 28 | UAE |

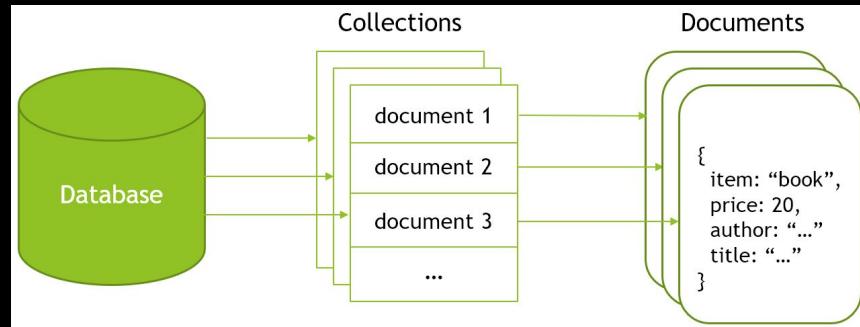
- **Relational Model Use of SQL:** Utilize **SQL** for querying and managing data, which is a standardized and widely-used language.
- **ACID Compliance:** Support transactions that are Atomic, Consistent, Isolated, and Durable.
- **Complex Queries:** Excel at handling complex queries and relationships between data.

```
SELECT age, country  
FROM Customers  
WHERE country = 'USA';
```

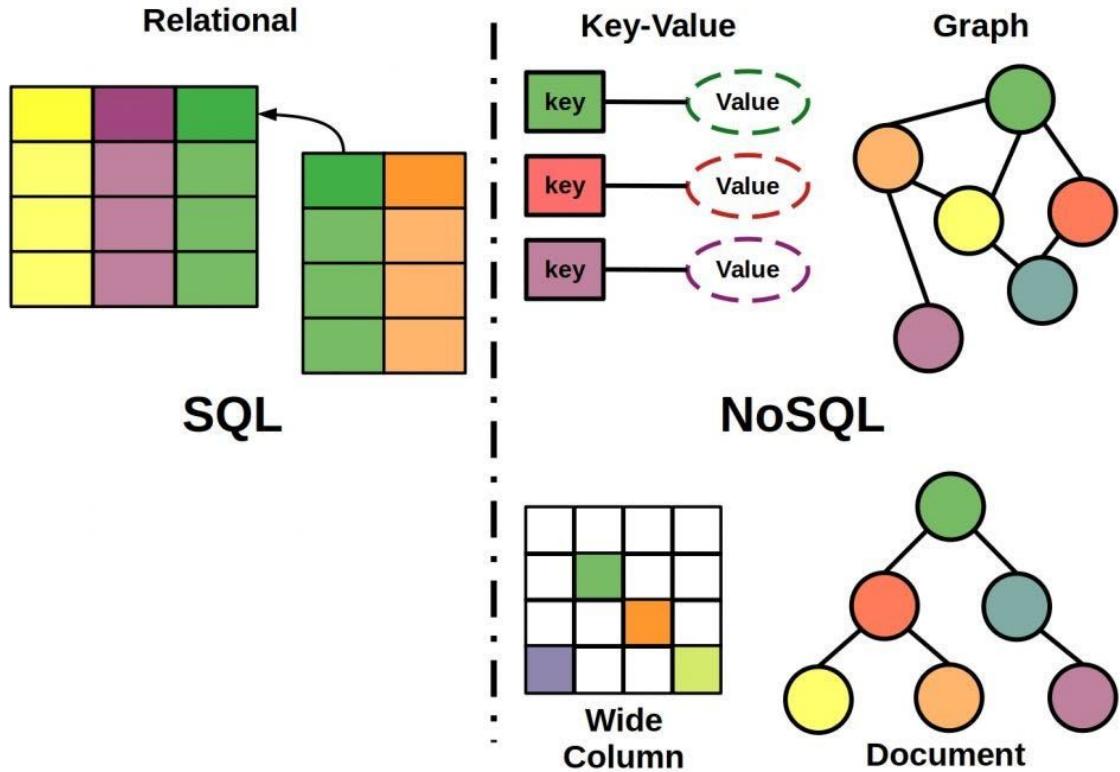
| age | country |
|-----|---------|
| 31 | USA |
| 22 | USA |

15.3 Introduction to NoSQL DB

- **Flexible Schema:** Allow for **dynamic schemas**, accommodating unstructured or semi-structured data without predefined structures.
- **Duplicacy over Relations:** Duplicates data across records (**denormalization**) to enhance performance and scalability, rather than relying on **complex relationships** and joins as in relational databases.
- **Horizontal Scalability:** Designed to **scale out** by adding more servers, handling large volumes of data efficiently.
- **Performance:** Optimized for **high throughput** and **low latency**, suitable for real-time applications.



15.4 SQL vs NoSQL



15.4 SQL vs NoSQL

| Feature | SQL Databases | NoSQL Databases |
|-----------------|---|---|
| Data Model | Relational (tables with rows and columns) | Non-relational (document, key-value, graph, etc.) |
| Schema | Fixed schema (predefined structure) | Flexible schema (dynamic structure) |
| Scalability | Vertically scalable (scale up) | Horizontally scalable (scale out) |
| Query Language | SQL (Structured Query Language) | Various query languages and APIs |
| ACID Compliance | Strong ACID compliance | Varies; often prioritizes performance over ACID |
| Use Cases | Structured data and complex queries | Unstructured data and real-time applications |

15.5 Installing MySQL

MySQL™

dev.mysql.com/downloads/mysql/ Guest (4) ...

MySQL Community Downloads

MySQL Enterprise Edition for Developers
Free for learning, developing, and prototyping.
[Download Now »](#)

MySQL Community Server

General Availability (GA) Releases Archives ⓘ

MySQL Community Server 9.1.0 Innovation

Select Version:
9.1.0 Innovation

Select Operating System:
Microsoft Windows

| Version | File Type | Size | Action |
|---------|---|--------|---|
| 9.1.0 | Windows (x86, 64-bit), MSI Installer | 118.1M | Download |
| | (mysql-9.1.0-winx64.msi) | | MD5: 7a26420bb3446eab56f389dba05a9718 Signature |
| 9.1.0 | Windows (x86, 64-bit), ZIP Archive | 288.4M | Download |
| | (mysql-9.1.0-winx64.zip) | | MD5: 0a2333afc4ef07471bda89232697698e Signature |
| 9.1.0 | Windows (x86, 64-bit), ZIP Archive Debug Binaries & Test Suite | 822.6M | Download |
| | (mysql-9.1.0-winx64-debug-test.zip) | | MD5: cb0327a504fc8d983f98c0cbf451a31d Signature |

Tip: We suggest that you use the [MD5 checksums and GnuPG signatures](#) to verify the integrity of the packages you download.

15.5 Installing MySQL

MySQL™

dev.mysql.com/downloads/workbench/ Guest (4)

MySQL Community Downloads

MySQL Workbench

General Availability (GA) Releases Archives

MySQL Workbench 8.0.38

Select Operating System: Microsoft Windows

Recommended Download:

MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

Go to Download Page >

Other Downloads:

| Windows (x86, 64-bit), MSI Installer | 8.0.38 | 41.7M | Download |
|---|--------|-------|----------|
| (mysql-workbench-community-8.0.38-winx64.msi) | | | |

MD5: 30ea58c9f40816566ac4cccd2f136f1e2 | Signature

We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

ORACLE © 2024 Oracle

Privacy / Do Not Sell My Info | Terms of Use | Trademark Policy | Cookie Preferences

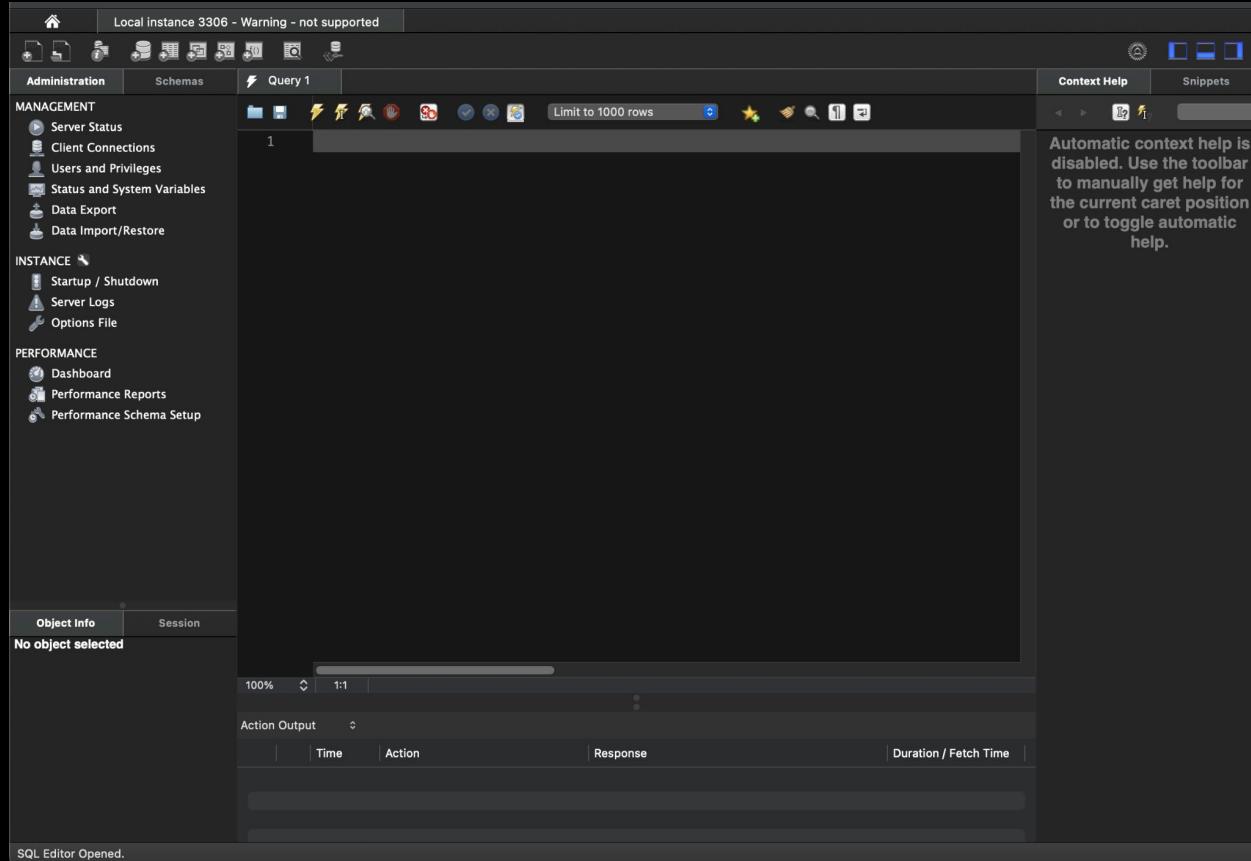
15.5 Installing MySQL

MySQL™

The screenshot shows the MySQL Workbench application window. The title bar reads "MySQL Workbench". On the left, there is a vertical toolbar with icons for file operations, a connection manager, schema browser, and query editor. The main content area has a dark background with the text "Welcome to MySQL Workbench" in large white letters. Below this, a paragraph describes the tool's features: "MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database." At the bottom of the main area, there are three links: "Browse Documentation >", "Read the Blog >", and "Discuss on the Forums >". Below these links is a section titled "MySQL Connections" with a plus sign icon and a refresh icon. A search bar labeled "Filter connections" is located next to it. A connection entry for "Local instance 3306" is listed, showing a user icon, the name "root", and the address "localhost:3306". The status bar at the bottom left says "Ready."

15.5 Installing MySQL

MySQL™



15.5 Installing MySQL

MySQL™

MySQL Workbench

Local Instance 3306 - Warning - not supported

Administration Schemas Query 1 new_schema - Schema Context Help Snippets

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Schema Editor

Specify the name of the schema here. You can use any combination of ANSI letters, numbers underscore character for names that don't require quoting. For more flexibility you can use the Unicode Basic Multilingual Plane (BMP), but you will have to quote the name later when you refer to it.

Schema Name: Rename References

The character set and its collation selected here will be used when no other charset/collation database object (it uses the DEFAULT value then). Setting DEFAULT here will make the schema server defaults.

Character Set: Default Charset

Collation: Default Collation

Object Info Session

No object selected

Action Output

| Time | Action | Response | Duration / Fetch Time |
|------|--------|----------|-----------------------|
| | | | |

SQL Editor Opened.

This screenshot shows the MySQL Workbench interface. The main window is titled 'Schema Editor' and displays a form for creating a new schema. The 'Schema Name' field contains 'new_schema'. Below it, a note states: 'The character set and its collation selected here will be used when no other charset/collation database object (it uses the DEFAULT value then). Setting DEFAULT here will make the schema server defaults.' There are dropdown menus for 'Character Set' (set to 'Default Charset') and 'Collation' (set to 'Default Collation'). The left sidebar has sections for 'MANAGEMENT', 'INSTANCE', and 'PERFORMANCE', each with several sub-options. The bottom left shows 'Object Info' and 'Session' tabs, with 'Object Info' active. The status bar at the bottom says 'SQL Editor Opened.'

15.6 Connecting App to DB

The screenshot shows a terminal window and a code editor side-by-side.

In the terminal window, the command `npm install --save mysql2` was run, resulting in:

```
prashantjain@Prashants-Mac-mini Chapter 13 - MVC % npm install --save mysql2
added 12 packages, and audited 222 packages in 586ms
49 packages are looking for funding
  run `npm fund` for details
```

The code editor displays the file `database.js` with the following content:

```
utils > database.js > ...
1 const mysql = require("mysql2");
2
3 const pool = mysql.createPool({
4   host: "localhost",
5   user: "root",
6   password: "CompleteCoding@01",
7   database: "airbnb",
8 });
9
10 module.exports = pool.promise();
```

15.7 Creating homes Table

Query 1 homes - Table Schema: airbnb

Name: homes Schema: airbnb

Column Datatype PK NN UQ B... UN ZF AI G Default / Expression

| Column | Datatype | PK | NN | UQ | B... | UN | ZF | AI | G | Default / Expression |
|-----------------|--------------|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|-------------------------------------|--------------------------|--------------------------|-------------------------------------|----------------------|
| id | INT | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | |
| name | VARCHAR(255) | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| price | DOUBLE | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| description | LONGTEXT | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| imageUrl | VARCHAR(255) | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| location | VARCHAR(45) | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| rating | DOUBLE | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | |
| <click to edit> | | | | | | | | | | |

Column details "

Column Name:

Charset/Collation: Default Charset Default Collation

Comments:

Datatype:

Expression:

Storage: VIRTUAL STORED

Primary Key Not NULL Unique

Binary Unsigned ZeroFill

Auto Increment Generated

Columns Indexes Foreign Keys Triggers Partitioning Options Apply Revert

15.7 Creating homes Table

```
- CREATE TABLE `airbnb`.`homes` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `price` DOUBLE NOT NULL,
  `description` LONGTEXT NOT NULL,
  `imageUrl` VARCHAR(255) NOT NULL,
  `location` VARCHAR(45) NOT NULL,
  `rating` DOUBLE NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE);
```

15.7 Creating homes Table

The screenshot shows the MySQL Workbench interface. The top navigation bar includes tabs for Administration, Schemas, Query 3, and the current tab, homes. Below the navigation is a toolbar with various icons for database management. The left sidebar lists SCHEMAS (airbnb) and TABLES (homes). The 'homes' table is selected, showing options like Columns, Indexes, Foreign Keys, Triggers, Views, Stored Procedures, Functions, and sys. The main workspace displays a Result Grid for the query `SELECT * FROM airbnb.homes;`. The grid has columns for id, name, price, description, imageUrl, location, and rating, all currently showing NULL values. A sidebar on the right provides links to Result Grid, Form Editor, Field Types, and a script editor.

Apply SQL Script to Database

Review the SQL Script to be Applied on the Database

Please review the following SQL script that will be applied to the database.

Note that once applied, these statements may not be revertible without losing some of the data.

You can also manually change the SQL statements before execution.

```
1    INSERT INTO `airbnb`.`homes` (`name`, `price`, `description`, `imageUrl`, `location`) VALUES ('Utsav', '999', 'the best holiday home', '/images/house1.png', 'delhi');
2
```

15.8 Querying homes in App

```
const db = require("./utils/database");

db.execute("SELECT * FROM homes").then(([rows, fields]) => {
  console.log(rows);
  console.log(fields);
}).catch((error) => {
  console.log("Error Fetching Homes", error);
});
```

```
Server running on address http://localhost:3000
[
  {
    id: 1,
    name: 'Utsav',
    price: 999,
    description: 'the best holiday home',
    imageUrl: '/images/house1.png',
    location: 'delhi',
    rating: 4.5
  }
]
`id` INT UNSIGNED NOT NULL PRIMARY KEY UNIQUE_KEY AUTO_INCREMENT,
`name` VARCHAR(255) NOT NULL,
`price` DOUBLE NOT NULL,
`description` LONGTEXT NOT NULL,
`imageUrl` VARCHAR(255) NOT NULL,
`location` VARCHAR(45) NOT NULL,
`rating` DOUBLE NOT NULL
]
```

15.9 Adding DB in Models

1. Remove the test code from app.js
2. Change the Home.js model file to remove all code related to file operations.
3. Import the DB from the utils.
4. Change photoUrl to imageUrl and houseName to name in the entire project.
5. Implement fetchAll:
 - a. Using the query we used while testing.
 - b. fetchAll will not take a callback but return a promise.
6. Go to StoreController and use the promise to get the data here.
7. Fix all the usages of fetchAll.

15.9 Adding DB in Models

2,3.

```
const db = require("../utils/database");
const Favourites = require("./favourites");

module.exports = class Home {
  constructor(houseName, price, location, rating, photoUrl) {
    this.houseName = houseName;
    this.price = price;
    this.location = location;
    this.rating = rating;
    this.photoUrl = photoUrl;
  }

  save() { ↗ tab
  }

  static fetchAll() {
  }

  static findById(id) {
  }

  static deleteById(id) {
  }
};
```

15.9 Adding DB in Models

```
5. static fetchAll() {  
    return db.execute("SELECT * FROM homes");  
}  
  
6. exports.getHomes = (req, res, next) => {  
    Home.fetchAll()  
        .then(([rows, fields]) => {  
            res.render("store/home-list", {  
                registeredHomes: rows,  
                pageTitle: "Homes List",  
                currentPage: "Home",  
            });  
        })  
        .catch((error) => {  
            console.log("Error Fetching Homes", error);  
        });  
};
```

15.9 Adding DB in Models

7.

```
exports.getIndex = (req, res, next) => {
  Home.fetchAll()
    .then(([rows, fields]) => {
      res.render("store/index", {
        registeredHomes: rows,
        pageTitle: "airbnb Home",
        currentPage: "index",
      })
    })
    .catch((error) => {
      console.log("Error Fetching Homes", error);
    });
};

exports.getHostHomes = (req, res, next) => {
  Home.fetchAll().then(([rows, fields]) => {
    res.render("host/host-home-list", {
      registeredHomes: rows,
      pageTitle: "Host Homes",
      currentPage: "hostHomes",
    });
  }).catch((error) => {
    console.log("Error Fetching Homes", error);
  });
};
```

```
exports.getFavouriteList = (req, res, next) => {
  Favourites.getFavourites((favourites) => {
    Home.fetchAll().then(([registeredHomes, fields]) => {
      const favouritesWithDetails = favourites.map((homeId) =>
        registeredHomes.find((home) => home.id === homeId)
      );
      res.render("store/favourite-list", {
        favourites: favouritesWithDetails,
        pageTitle: "My Favourites",
        currentPage: "favourites",
      });
    }).catch((error) => {
      console.log("Error Fetching Homes", error);
    });
  });
};
```

15.10 Adding Home in Model

1. Add the **description** field in home. Change constructor and usage.
2. Make changes in UI to input and show it everywhere.
3. Implement the save method using the insert query.
4. Change the usages of save method to use the promise.

15.10 Adding Home in Model

1.

```
module.exports = class Home {  
    constructor(name, description, price, location, rating, imageUrl) {  
        this.name = name;  
        this.description = description;  
        this.price = price;  
        this.location = location;  
        this.rating = rating;  
        this.imageUrl = imageUrl;  
    }  
  
    exports.postAddHome = (req, res, next) => {  
        const { name, description, price, location, rating, imageUrl } = req.body;  
        const home = new Home(name, description, price, location, rating, imageUrl);  
  
        exports.postEditHome = (req, res, next) => {  
            const { id, name, description, price, location, rating, imageUrl } = req.body;  
            const home = new Home(name, description, price, location, rating, imageUrl);  
        }  
    }  
}
```

15.10 Adding Home in Model

2.

```
<input  
    type="text"  
    name="description"  
    value="<% home ? home.description : '' %>"  
    placeholder="Enter your House Description"  
    class="w-full px-4 py-2 mb-4 border rounded-md focus:outline-none focus:ring-2  
    focus:ring-red-500"/>  
  
<div class="border-b pb-4">  
    <h3 class="text-2xl font-semibold mb-2">Description</h3>  
    <p class="text-gray-600"><% home.description %></p>  
</div>  
  
<div class="border-b pb-4">  
    <h3 class="text-2xl font-semibold mb-2">Location</h3>  
    <p class="text-gray-600"><% home.location %></p>  
</div>
```

15.10 Adding Home in Model

3.

```
save() {
  return db.execute(
    "INSERT INTO homes (name, price, location, rating, imageUrl) VALUES (?, ?, ?, ?, ?)",
    [this.name, this.price, this.location, this.rating, this.imageUrl]
  );
}
```

4.

```
exports.postAddHome = (req, res, next) => {
  const { name, description, price, location, rating, imageUrl } = req.body;
  const home = new Home(name, description, price, location, rating, imageUrl);
  home.save().then(() => {
    res.render("host/home-added", {
      pageTitle: "Home Added Successfully",
      currentPage: "homeAdded",
    });
  }).catch((error) => {
    console.log("Error Adding Home", error);
  });
};

exports.postEditHome = (req, res, next) => {
  const { id, name, description, price, location, rating, imageUrl } = req.body;
  const home = new Home(name, description, price, location, rating, imageUrl);
  home.id = id;
  home.save().then(() => {
    res.redirect("/host/host-home-list");
  }).catch((error) => {
    console.log("Error Editing Home", error);
  });
};
```

15.11 Implementing Model using Where

```
static findById(id) {
  return db.execute("SELECT * FROM homes WHERE id = ?", [id]);
}

static deleteById(id) {
  return db.execute("DELETE FROM homes WHERE id = ?", [id]);
}

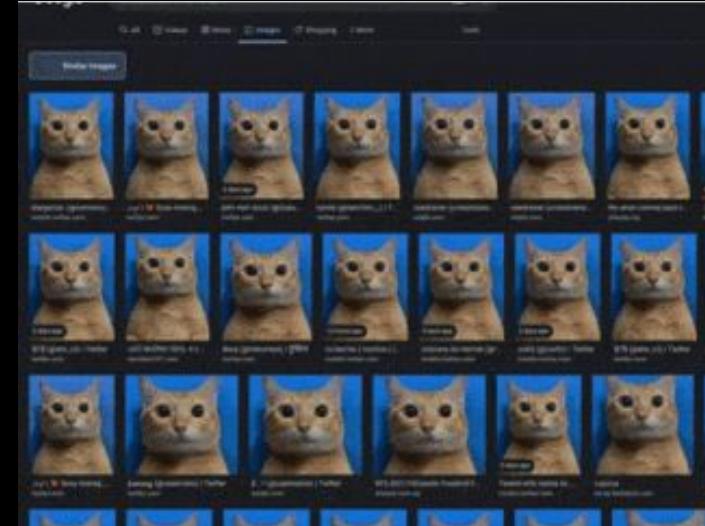
exports.postDeleteHome = (req, res, next) => {
  const homeId = req.params.homeId;
  Home.deleteById(homeId).then(() => {
    res.redirect("/host/host-home-list");
  }).catch((error) => {
    console.log("Error Deleting Home", error);
  });
};
```

```
exports.getHome = (req, res, next) => {
  const homeId = req.params.homeId;
  Home.findById(homeId).then(([rows]) => {
    const home = rows[0];
    if (!home) {
      return res.redirect("/homes");
    }
    res.render("store/home-detail", {
      home: home,
      pageTitle: home.name,
      currentPage: "homes",
    });
  }).catch((error) => {
    console.log("Error Fetching Home", error);
  });
};
```



16. Introduction to NoSQL

1. What is MongoDB
2. Setting up MongoDB
3. Installing MongoDB Driver
4. Creating MongoDB Connection
5. Saving a Home
6. Install MongoDB Compass
7. Install MongoDB for VSCode
8. Fetching all Homes
9. Fetching one Home





16.1 What is MongoDB

1. MongoDB is the **product** and the **company** that builds it.
2. The name comes from the work **Humongous**.
3. **NoSQL Document Database:** Stores data in flexible, JSON-like documents.
4. **Dynamic Schema:** Allows **fields** to vary across documents without predefined schemas.
5. **High Performance:** Optimized for **fast** read and write operations.
6. **Scalability:** Supports **horizontal scaling** through sharding.
7. **High Availability:** Provides replication with automatic failover.
8. **Rich Query Capabilities:** Offers **powerful** querying, indexing, and aggregation.
9. **Geospatial and Text Search:** Includes support for location-based and full-text queries.
10. **Cross-Platform Compatibility:** Works with various **operating systems** and programming languages.
11. **Easy Integration:** Integrates smoothly with modern development stacks.

```
{  
  "_id": "4f5b5c85-d8d3-4f58-8acf-3f5e5e4e59ea",  
  "Items": [  
    {  
      "ProductId": 1,  
      "ProductName": "Elden Ring",  
      "Price": "49.97",  
      "Quantity": 1  
    },  
    {  
      "ProductId": 2,  
      "ProductName": "FIFA 23",  
      "Price": "69.97",  
      "Quantity": 1  
    }  
  ]  
}
```



MongoDB®



16.2 Setting up MongoDB

MongoDB Community Server

Download

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

The database is also offered as a fully-managed service with [MongoDB Atlas](#). Get access to advanced functionality such as auto-scaling, serverless instances, full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.

[Give it a try with a free, highly-available 512 MB cluster.](#) or get started from your terminal with the following two commands:

```
$ brew install mongodb-atlas  
$ atlas setup
```

Version
8.0.3 (current)

Platform
Windows x64

Package
msi

[Download](#)

[Copy link](#)

More Options

VS

Try MongoDB Atlas

A developer data platform built around a fully managed MongoDB service. Address transactional, search, and analytical workloads.

[Explore all our products](#)

Create your Atlas Account

The multi-cloud developer data platform.

First Name*

Last Name*

Company



16.2 Setting up MongoDB

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

| | | |
|--|-------------|------------|
| <input type="radio"/> M10 | \$0.08/hour | |
| Dedicated cluster for development environments and low-traffic applications. | | |
| STORAGE | RAM | vCPU |
| 10 GB | 2 GB | 2 vCPUs |
| <input type="radio"/> Serverless | | |
| For application development and testing, or workloads with variable traffic. | | |
| STORAGE | RAM | vCPU |
| Up to 1TB | Auto-scale | Auto-scale |
| <input checked="" type="radio"/> M0 | Free | |
| For learning and exploring MongoDB in a cloud environment. | | |
| STORAGE | RAM | vCPU |
| 512 MB | Shared | Shared |

Free forever! Your free cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

| | |
|--|---|
| Configurations | Quick setup |
| Name You cannot change the name once the cluster is created. <input type="text" value="KGCluster"/> | <input checked="" type="checkbox"/> Automate security setup <small>i</small> <input checked="" type="checkbox"/> Preload sample dataset <small>i</small> |
| Provider AWS Google Cloud Azure | |
| Region Mumbai (ap-south-1) ★ Low carbon emissions <small>i</small> | |
| Tag (optional) Create your first tag to categorize and label your resources; more tags can be added later. Learn more . | |
| <input type="text" value="Select or enter key"/> : <input type="text" value="Select or enter value"/> | |



16.2 Setting up MongoDB

Connect to KGCluster

1 Set up connection security 2 Choose a connection method 3 Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more ↗](#)

1. Add a connection IP address

Your current IP address (160.202.37.194) has been added to enable local connectivity. Only an IP address you add to your Access List will be able to connect to your project's clusters. Add more later in [Network Access ↗](#).

2. Create a database user

This first user will have [atlasAdmin ↗](#) permissions for this project.

We autogenerated a username and password. You can use this or create your own.

i You'll need your database user's credentials in the next step. Copy the database user password.

Username root **Password** root **HIDE** **Copy**

Create Database User

Close **Choose a connection method**



16.2 Setting up MongoDB

Connect to KGCluster

Set up connection security ✓

Choose a connection method 2

Connect 3

Connect to your application

Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

Compass
Explore, modify, and visualize your data with MongoDB's GUI

Shell
Quickly add & update data using MongoDB's Javascript command-line interface

MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment

Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization

[Go Back](#) [Close](#)

Connect to KGCluster

Set up connection security ✓

Choose a connection method ✓

Connect 3

Connecting with MongoDB Driver

1. Select your driver and version
We recommend installing and using the latest driver version.

| Driver | Version |
|---------|--------------|
| Node.js | 5.5 or later |

2. Install your driver
Run the following on the command line
`npm install mongodb`

[View MongoDB Node.js Driver installation instructions.](#)

3. Add your connection string into your application code

 **KGCluster is provisioning...**

It takes an average of 10-15 seconds to provision your deployment. Clusters are built with 3 nodes for resiliency.

You will be able to copy your connection string shortly. If you would like to come back to this later, you can go to the [Atlas Home Page](#).

RESOURCES

[Get started with the Node.js Driver](#) [Node.js Starter Sample App](#) [Access your Database Users](#) [Troubleshoot Connections](#)

[Go Back](#) [Done](#)



16.2 Setting up MongoDB

Project 0 Data Services Charts

Overview We are deploying your changes (current action: creating a plan)

DATABASE PRASHANT'S ORG - 2024-11-08 > PROJECT 0

Clusters

SERVICES

Atlas Search IP Access List Peering Private Endpoint

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

Advanced

New On Atlas 7

Goto

The screenshot shows the MongoDB Cloud interface for a project named 'PRASHANT'S ORG - 2024-11-08 > PROJECT 0'. The 'Data Services' tab is active. In the 'Network Access' section, the 'IP Access List' tab is selected. A note states: 'You will only be able to connect to your cluster from the following list of IP Addresses:' followed by the IP address '160.202.37.194/32 (includes your current IP address)'. The 'Network Access' link in the sidebar is highlighted with a red box.



16.3 Installing MongoDB Driver

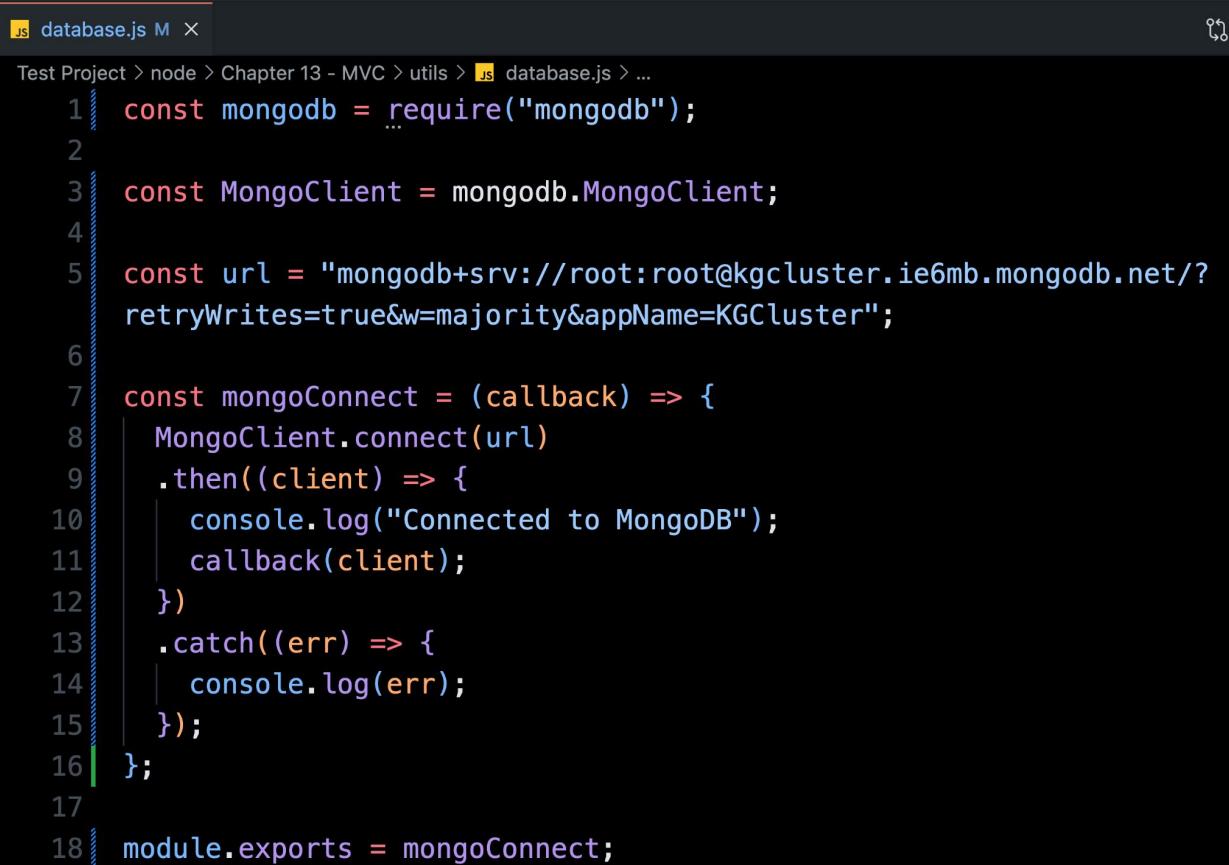
✓ TERMINAL

```
● prashantjain@Prashants-Mac-mini Chapter 13 - MVC % npm install mongodb
added 12 packages, and audited 234 packages in 2s
49 packages are looking for funding
  run `npm fund` for details
2 low severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
○ prashantjain@Prashants-Mac-mini Chapter 13 - MVC %
```

node 16.3 Installing MongoDB Driver



```
JS database.js M × ⚙
Test Project > node > Chapter 13 - MVC > utils > JS database.js > ...
1 const mongodb = require("mongodb");
2
3 const MongoClient = mongodb.MongoClient;
4
5 const url = "mongodb+srv://root:root@kgcluster.ie6mb.mongodb.net/?retryWrites=true&w=majority&appName=KGCluster";
6
7 const mongoConnect = (callback) => {
8   MongoClient.connect(url)
9     .then((client) => {
10       console.log("Connected to MongoDB");
11       callback(client);
12     })
13     .catch((err) => {
14       console.log(err);
15     });
16   };
17
18 module.exports = mongoConnect;
```

node 16.3 Installing MongoDB Driver

```
Test Project > node > Chapter 13 - MVC > app.js > ...  
25  
26 | const mongoConnect = require("./utils/database");  
27 | const PORT = 3000;  
28 | mongoConnect((client) => {  
29 |   console.log(client);  
30 |   app.listen(PORT, () => {  
31 |     console.log(`Server running on address 

```
Connected to MongoDB
<ref *1> MongoClient {
 _events: \[Object: null prototype\] {},
 _eventsCount: 0,
 _maxListeners: undefined,
 mongoLogger: undefined,
 s: {
 url: 'mongodb+srv://root:root@kgcluster.ie6mb.mongodb.net/?retryWrites=true&w=majority&appName=KGCluster',
 bsonOptions: {
 raw: false,
 useBigInt64: false,
 promoteLongs: true,
 promoteValues: true,
 promoteBuffers: false,
 ignoreUndefined: false,
 bsonRegExp: false,
 serializeFunctions: false,
 fieldsAsRaw: {},
 enableUtf8Validation: true
 },
 namespace: MongoDBNamespace { db: 'admin', collection: undefined },
 hasBeenClosed: false,
 sessionPool: ServerSessionPool { client: \[Circular *1\], sessions: \[List\] },
 activeSessions: Set\(0\) {},
 authProviders: MongoClientAuthProviders { existingProviders: \[Map\] },
 options: \[Getter\],
 readConcern: \[Getter\],
 writeConcern: \[Getter\],
 readPreference: \[Getter\],
 isMongoClient: \[Getter\]
 },
 connectionLock: undefined,
 topology: Topology {
 _events: \[Object: null prototype\] {
 topologyDescriptionChanged: \[Array\],
 connectionPoolCreated: \[Function \(anonymous\)\]
 }
 }
}
```


```

```
'  
Server running on address http://localhost:3000  
TypeError: db.execute is not a function  
  at Home.fetchAll (/Users/prashantjain/workspace/stuff/Test Project/node/Chapter 13 - MVC/app.js:2:15)  
    at exports.getIndex (/Users/prashantjain/workspace/stuff/Test Project/node/Chapter 13 - MVC/app.js:5:8)  
      at Layer.handle [as handle_request] (/Users/prashantjain/workspace/stuff/Test Project/node_modules/express/lib/router/layer.js:95:5)  
        at next (/Users/prashantjain/workspace/stuff/Test Project/node/Chapter 13 - MVC/app.js:149:13)  
          at Route.dispatch (/Users/prashantjain/workspace/stuff/Test Project/node_modules/express/lib/router/route.js:119:3)  
            at Layer.handle [as handle_request] (/Users/prashantjain/workspace/stuff/Test Project/node_modules/express/lib/router/layer.js:95:5)  
              at /Users/prashantjain/workspace/stuff/Test Project/node/Chapter 13 - MVC/app.js:284:15  
                at Function.process_params (/Users/prashantjain/workspace/stuff/Test Project/node_modules/express/lib/router/index.js:346:12)  
                  at next (/Users/prashantjain/workspace/stuff/Test Project/node/Chapter 13 - MVC/app.js:280:10)  
                    at Function.handle (/Users/prashantjain/workspace/stuff/Test Project/node_modules/express/lib/router/index.js:175:3)
```



16.3 Installing MongoDB Driver

```
module.exports = class Home {  
  constructor(name, description, price, location, rating,  
    imageUrl) {  
    this.name = name;  
    this.description = description;  
    this.price = price;  
    this.location = location;  
    this.rating = rating;  
    this.imageUrl = imageUrl;  
  }  
  
  save() {  
  }  
  
  static fetchAll() {  
  }  
  
  static findById(id) {  
  }  
  
  static deleteById(id) {  
  }  
};
```



16.4 Creating MongoDB Connection

```
database.js M X app.js M home.js M
Test Project > node > Chapter 13 - MVC > utils > database.js > ...
1 const mongodb = require("mongodb");
2
3 const MongoClient = mongodb.MongoClient;
4
5 const url = "mongodb+srv://root:root@kgcluster.ie6mb.mongodb.net/?retryWrites=true&
w=majority&appName=KGCluster";
6
7 let _db;
8
9 const mongoConnect = (callback) => {
10   MongoClient.connect(url)
11     .then((client) => {
12       console.log("Connected to MongoDB");
13       _db = client.db("airbnb");
14       callback();
15     })
16     .catch((err) => {
17       console.log(err);
18       throw err;
19     });
20 };
21
22 const getDb = () => {
23   if (!_db) {
24     throw new Error("Database not connected");
25   }
26   return _db;
27 };
28
29 exports.mongoConnect = mongoConnect;
30 exports.getDb = getDb;
```



16.5 Saving a Home

```
save() {
  const db = getDb();
  // insert Many takes an array of objects
  return db.collection("homes").insertOne(this).then((result) => {
    console.log(result);
  });
}

{
  acknowledged: true,
  insertedId: new ObjectId('672df86c3d20696a2b523c79')
}
```

localhost:3000/host/add-home

airbnb Homes-List Favourites Bookings Host Homes Add Home

Register Your Home on AirBnB

| |
|--------------------|
| Utsav |
| the best house |
| 1999 |
| ghaziabad |
| 4.5 |
| /images/house1.png |

Add Home

16.6 Install MongoDB Compass

[Products](#) ▾[Resources](#) ▾[Solutions](#) ▾[Company](#) ▾[Pricing](#)[Eng](#) ▾[Support](#)[Sign In](#)[Try Free](#)[TOOLS](#)

Compass. The GUI for MongoDB.

Compass is a free interactive tool for querying, optimizing, and analyzing your MongoDB data. Get key insights, drag and drop to build pipelines, and more.

[Download Now](#)[Read the docs →](#)

16.6 Install MongoDB Compass

[Learn more](#)

Version

1.44.6 (Stable)



Platform

Windows 64-bit (10+)



Package

exe



[Download ↓](#)



[Copy link](#)

More Options

16.6 Install MongoDB Compass

The screenshot shows the MongoDB Compass application window. The title bar reads "MongoDB Compass". The menu bar includes "Connections", "Edit", "View", and "Help". The left sidebar is titled "Compass" and contains sections for "My Queries" and "CONNECTIONS". The "CONNECTIONS" section displays the message "You have not connected to any deployments." and features a green button labeled "+ Add new connection". The main content area is titled "Welcome to MongoDB Compass" with the sub-instruction "To get started, connect to an existing server or". It features a large icon of a magnifying glass over a cloud. A red box highlights the green "+ Add new connection" button. Below it, a light blue box contains the text "New to Compass and don't have a cluster? If you don't already have a cluster, you can create one for free using MongoDB Atlas" and a "CREATE FREE CLUSTER" button. The bottom of the screen shows a taskbar with various icons, including a currency converter showing "USD/CNY +0.32%", a search bar, and several application icons.

16.6 Install MongoDB Compass

New Connection

Manage your connection settings

URI [i](#) Edit Connection String

```
mongodb+srv://root:*****@kgcluster.ie6mb.mongodb.net/
```

Name **Color** ▾

Favorite this connection
Favoriting a connection will pin it to the top of your list of connections

[Advanced Connection Options](#)

[Cancel](#) [Save](#) [Save & Connect](#)

How do I find my connection string in Atlas?
If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.
[See example ↗](#)

How do I format my connection string?
[See example ↗](#)

16.6 Install MongoDB Compass

The screenshot shows the MongoDB Compass application interface. On the left, there's a sidebar titled "Compass" with sections for "My Queries" and "CONNECTIONS". Under "CONNECTIONS", a tree view shows a connection named "kgcluster.ie6mb.mongodb.net" which has four databases: "admin", "airbnb", "config", "local", and "sample_mflix". The "airbnb" database is currently selected and highlighted with a blue border. The main workspace displays the "airbnb" connection details. At the top, there are tabs for "Welcome" and "airbnb", and a "+" button to add new connections. To the right of the tabs are buttons for "Open MongoDB shell", "Create collection", and "Refresh". Below these are sorting and filtering controls: "Sort by Collection Name" and a "View" dropdown menu with icons for list and grid. The central area shows the "homes" collection with the following statistics:

| Storage size: | Documents: | Avg. document size: | Indexes: | Total index size: |
|---------------|------------|---------------------|----------|-------------------|
| 20.48 kB | 1 | 159.00 B | 1 | 20.48 kB |

16.6 Install MongoDB Compass

The screenshot shows the MongoDB Compass interface for a database cluster named 'kgcluster'. The current view is on the 'homes' collection under the 'airbnb' database. The top navigation bar includes tabs for 'Welcome', 'homes', and a '+' button. On the right, there's a button to 'Open MongoDB shell'.

The main interface displays the 'Documents' tab, which contains one document. The document details are as follows:

```
_id: ObjectId('672df86c3d20696a2b523c79')
name : "Utsav"
description : "the best house"
price : "1999"
location : "ghaziabad"
rating : "4.5"
imageUrl : "/images/house1.png"
```

Below the document details, there are several action buttons: '+ ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. To the right, there are pagination controls showing '25' items, '1-1 of 1', and icons for navigating between documents.

16.7 Install MongoDB for VSCode

mongodb

MARKETPLACE 39

- MongoDB for VS ...** ⚡ 1.9M ★ 4.5
Connect to MongoDB and Atlas ...
mongodb [Install](#)
- ES7 JavaScript/Nod...** ⚡ 87K ★ 5
Simple extension for Node, java...
abrahamwilliam007 [Install](#)
- Auto MongoDB** ⚡ 23K ★ 3
One click to run MongoDB without...
AllenLawrence [Install](#)
- Azure Databases** ⚡ 1.9M ★ 3
Create, browse, and update glob...
ms-azuretools [Install](#)
- Mongodb-dly** ⚡ 7K ★ 5
It is a quickly administrator of m...
devlikeyou [Install](#)
- MongoDB ID Generat...** ⚡ 4K ★ 5
Generates MongoDB IDs
JonathanLewis [Install](#)
- MySQL** ⚡ 1.8M ★ 4
Database manager for MySQL/M...
cweijan [Install](#)
- Mongo Snippets f...** ⚡ 270K ★ 4.5
Provides snippets, boilerplate co...
roerohan [Install](#)
- Azure Tools** ⚡ 1.5M ★ 3
Get web site hosting, SQL and M...
A [Install](#)

MongoDB for VS Code v1.9.3

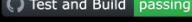
mongodb | ⚡ 1,900,000 | ★★★★★(39)

Connect to MongoDB and Atlas directly from your VS Code environment, navigate your databases...

[Install](#) Auto Update

[DETAILS](#) [FEATURES](#)

MongoDB for VS Code

 Test and Build passing

MongoDB for VS Code makes it easy to work with your data in MongoDB directly from your VS Code environment. MongoDB for VS Code is the perfect companion for MongoDB Atlas, but you can also use it with your self-managed MongoDB instances.



Features

Navigate your MongoDB Data

[Categories](#)

- Programming Languages
- Snippets
- Data Science
- AI
- Chat

[Resources](#)

- Marketplace
- Issues
- Repository
- License
- mongodb

[More Info](#)

| | |
|---------------|------------------------|
| Published | 2020-05-12, 01:46:46 |
| Last released | 2024-10-24, 18:53:28 |
| Identifier | mongodb.mongodb-vscode |

16.7 Install MongoDB for VSCode

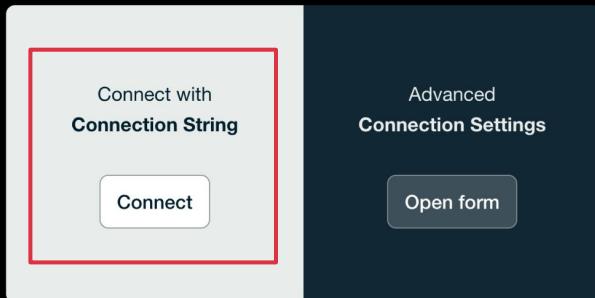


Navigate your databases and collections, use playgrounds for exploring and transforming
your data



Resources

● Not connected.



Cmd + Shift + P for all MongoDB Command Palette options



New to MongoDB and don't have a cluster?

Create one for free using [MongoDB Atlas](#).

Create free cluster

16.7 Install MongoDB for VSCode



Navigate your databases and collections, use playgrounds for exploring and transforming your data



Resources

Connected to: **kgcluster.ie6mb.mongodb.net**

All set. Ready to start?

Create a playground.

Create playground

Connect with
Connection String

Connect

Advanced
Connection Settings

Open form

Cmd + Shift + P for all MongoDB Command Palette options



New to MongoDB and don't have a cluster?

Create one for free using [MongoDB Atlas](#).

Create free cluster

16.7 Install MongoDB for VSCode

Currently connected to kgcluster.ie6mb.mongodb.net. Click here to change connection.

```
1 /* global use, db */
2 // MongoDB Playground
3 // To disable this template go to Settings | MongoDB | Use Default Template For Playground.
4 // Make sure you are connected to enable completions and to be able to run a playground.
5 // Use Ctrl+Space inside a snippet or a string literal to trigger completions.
6 // The result of the last command run in a playground is shown on the results panel.
7 // By default the first 20 documents will be returned with a cursor.
8 // Use 'console.log()' to print to the debug output.
9 // For more documentation on playgrounds please refer to
10 // https://www.mongodb.com/docs/mongodb-vscode/playgrounds/
11
12 // Select the database to use.
13 use('airbnb');
14
15 // Get all documents from homes collection
16 const homes = db.getCollection('homes').find({}).toArray();
17
18 // Print the results
19 console.log('All homes in collection:');
20 console.log(JSON.stringify(homes, null, 2));
21
22 // Print total count
23 const totalHomes = db.getCollection('homes').countDocuments();
24 console.log(`Total number of homes: ${totalHomes}`);
```

16.7 Install MongoDB for VSCode

Currently connected to kgcluster.ie6mb.mongodb.net. Click here to change connection.

```
1 /* global use, db */
2 // MongoDB Playground
3 // To disable this template go to Settings | MongoDB | Use Default Template For Playground.
4 // Make sure you are connected to enable completions and to be able to run a playground.
5 // Use Ctrl+Space inside a snippet or a string literal to trigger completions.
6 // The result of the last command run in a playground is shown on the results panel.
7 // By default the first 20 documents will be returned with a cursor.
8 // Use 'console.log()' to print to the debug output.
9 // For more documentation on playgrounds please refer to
10 // https://www.mongodb.com/docs/mongodb-vscode/playgrounds/
11
12 // Select the database to use.
13 use('airbnb');
14
15 // Get all documents from homes collection
16 const homes = db.getCollection('homes').find({}).toArray();
17
18 // Print the results
19 console.log('All homes in collection:');
20 console.log(JSON.stringify(homes, null, 2));
21
22 // Print total count
23 const totalHomes = db.getCollection('homes').countDocuments();
24 console.log(`Total number of homes: ${totalHomes}`);
```

```
All homes in collection:
[
  {
    "_id": "672df86c3d20696a2b523c79",
    "name": "Utsav",
    "description": "the best house",
    "price": "1999",
    "location": "ghaziabad",
    "rating": "4.5",
    "imageUrl": "/images/house1.png"
  }
]
Total number of homes: 1
```

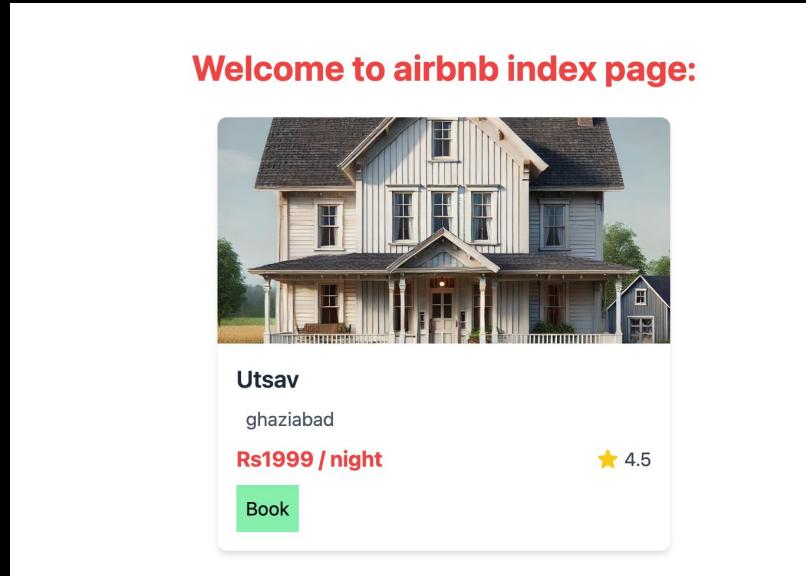
16.8 Fetching all Homes

```
static fetchAll() {  
  const db = getDb();  
  return db.collection("homes")  
    .find()  
    .toArray()  
    .then((homes) => {  
      console.log(homes);  
      return homes;  
    }).catch((err) => {  
      console.log(err);  
    });  
}
```

1. **Basic Usage:** `collection.find(query)` retrieves documents matching the query criteria.
2. **Returns a Cursor:** The method returns a cursor, an iterator over the result set.

16.8 Fetching all Homes

```
exports.getHomes = (req, res, next) => {
  Home.fetchAll()
    .then(rows => {
      res.render("store/home-list", {
        registeredHomes: rows,
        pageTitle: "Homes List",
        currentPage: "Home",
      });
    })
    .catch((error) => {
      console.log("Error Fetching Homes", error);
    });
};
```



16.9 Fetching one Home

```
static findById(homeId) {  
  const db = getDb();  
  return db.collection("homes")  
    .find({ _id: homeId })  
    .next()  
    .then((home) => {  
      console.log(home);  
      return home;  
    }).catch((err) => {  
      console.log(err);  
    });  
}
```

```
exports.getHome = (req, res, next) => {  
  const homeId = req.params.homeId;  
  Home.findById(homeId).then(home => {  
    if (!home) {  
      return res.redirect("/homes");  
    }  
    res.render("store/home-detail", {  
      home: home,  
      pageTitle: home.name,  
      currentPage: "homes",  
    });  
  }).catch((error) => {  
    console.log("Error Fetching Home", error);  
  });  
};
```

16.9 Fetching one Home

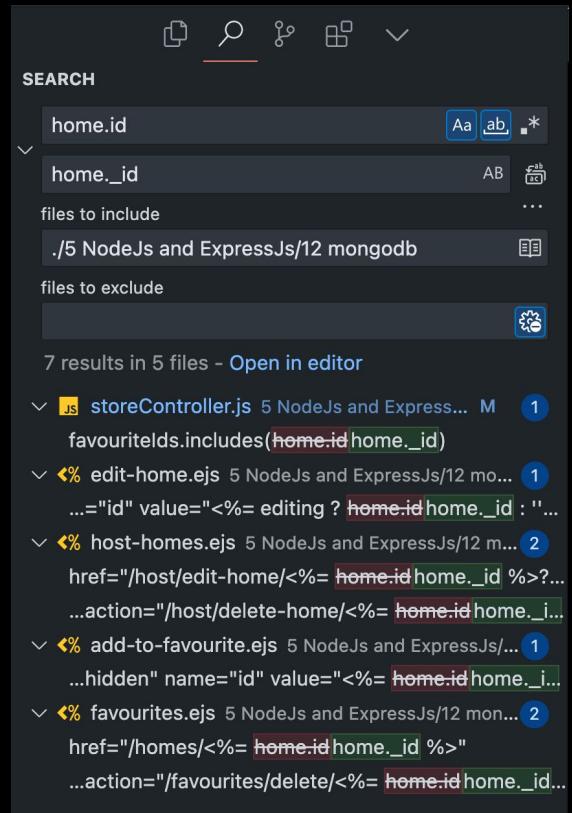
The screenshot shows a code editor interface with a search bar at the top. The search term 'home.id' is entered. Below the search bar, there is a dropdown menu with options: 'home._id' (which is currently selected), 'files to include', and 'files to exclude'. A 'clear' button (an 'X') is also present. The main area displays a list of 9 results found in 7 files. The results are listed under several files, each with a file icon, name, path, and a 'M' icon with a number indicating the count of matches.

| File | Path | Count |
|--------------------|---|-------|
| hostController.js | Test Project/node/Chapter 13 - MVC/controllers | 1 |
| storeController.js | Test Project/node/Chapter 13 - MVC/controllers | 1 |
| edit-home.ejs | Test Project/node/Chapter 13 - MVC/views/host | 1 |
| host-home-list.ejs | Test Project/node/Chapter 13 - MVC/views/host | 2 |
| favourite.ejs | Test Project/node/Chapter 13 - MVC/views/partials | 1 |
| favourite-list.ejs | Test Project/node/Chapter 13 - MVC/views/store | 2 |
| home-list.ejs | Test Project/node/Chapter 13 - MVC/views/store | 1 |

```
[  
  {  
    _id: new ObjectId('672df86c3d20696a2b523c79'),  
    name: 'Utsav',  
    description: 'the best house',  
    price: '1999',  
    location: 'ghaziabad',  
    rating: '4.5',  
    imageUrl: '/images/house1.png'  
  }  
]
```

16.9 Fetching one Home

```
static findById(homeId) {  
    const db = getDb();  
    return db.collection("homes")  
        .find({ _id: new ObjectId(String(homeId)) })  
        .next()  
        .then((home) => {  
            console.log(home);  
            return home;  
        }).catch((err) => {  
            console.log(err);  
        });  
}
```



16.9 Fetching one Home

Details of Utsav



Description
the best house

Location
ghaziabad

Price
\$1999 / night

Rating
★ 4.5 / 5

[Add to Favorites](#)

16.10 Supporting Edit & Delete

```
save() {  
  const db = getDb();  
  if (this.id) {  
    // Update existing home  
    return db  
      .collection("homes")  
      .updateOne(  
        { _id: new ObjectId(String(this.id)) },  
        { $set: this}  
      );  
  } else {  
    // Insert new home  
    return db  
      .collection("homes")  
      .insertOne(this)  
      .then((result) => {  
        console.log(result);  
      });  
  }  
}
```

```
static deleteById(homeId) {  
  const db = getDb();  
  return db  
    .collection("homes")  
    .deleteOne({ _id: new ObjectId(String(homeId)) });  
}
```

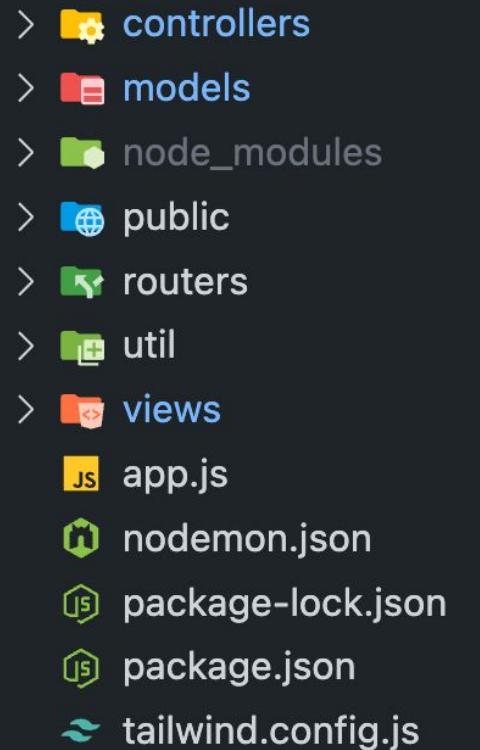
16.11 Adding MongoDB to Favourite

1. Remove all the file handling related code from **Favourite Model**.
2. Delete the data folder.
3. Change the following methods in **Favourite model** to use mongo:
 - a. `fetchAll`
 - b. `addToFavourites`
 - c. `deleteById`
4. Change the usages of **Favourite model** in **StoreController** to use the promise syntax
 - a. `getFavourites`
 - b. `postAddFavourites`
 - c. `postRemoveFavourite`

16.11 Adding MongoDB to Favourite

```
1. module.exports = class Favourite {  
  ...  
  
  static fetchAll() {  
  }  
  
  static addToFavourites(homeId) {  
  }  
  
  static deleteById(removeHomeId) {  
  }  
}
```

2.



```
> controllers  
> models  
> node_modules  
> public  
> routers  
> util  
> views  
  JS app.js  
  Nodemon nodemon.json  
  JS package-lock.json  
  JS package.json  
  Tailwind tailwind.config.js
```

16.11 Adding MongoDB to Favourite

```
3.a. static fetchAll() {  
    const db = getDb();  
    return db.collection("favourites")  
        .find()  
        .toArray()  
        .then(favourites => {  
            return favourites.map(favourite => favourite.homeId.toString());  
        });  
}
```

16.11 Adding MongoDB to Favourite

```
3.b. static addToFavourites(homeId) {  
    const db = getDb();  
    return db  
3.c.     .collection("favourites")  
     .insertOne({ homeId: new ObjectId(String(homeId)) })  
     .then(result => {  
         console.log(result);  
     });  
}  
  
static deleteById(removeHomeId) {  
    const db = getDb();  
    return db  
        .collection("favourites")  
        .deleteOne({ homeId: new ObjectId(String(removeHomeId)) });  
}
```

16.11 Adding MongoDB to Favourite

```
4.a. exports.getFavourites = (req, res, next) => {
    Favourite.fetchAll().then((favouriteIds) => {
        Home.fetchAll().then((registeredHomes) => {
            console.log(favouriteIds);
            console.log(registeredHomes);
            const favouriteHomes = registeredHomes.filter((home) =>
                favouriteIds.includes(home._id.toString())
            );
            res.render("store/favourites", {
                homes: favouriteHomes,
                pageTitle: "Favourites",
            });
        });
    });
};
```

16.11 Adding MongoDB to Favourite

4.b.

```
exports.postAddFavourites = (req, res, next) => {
  const homeId = req.body.id;
  Favourite.addToFavourites(homeId)
    .then(() => {
      res.redirect("/favourites");
    })
    .catch((err) => {
      console.log("Error while adding to favourites", err);
    });
};
```

16.11 Adding MongoDB to Favourite

4.c.

```
exports.postRemoveFavourite = (req, res, next) => {
  const homeId = req.params.homeId;
  Favourite.deleteById(homeId)
    .then(() => {
      res.redirect("/favourites");
    })
    .catch((err) => {
      console.log("Error while remove from favourites ", err);
    });
};
```



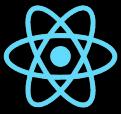
React

Practise Milestone

Take your **airbnb** forward:

- Change the favourite model to fix the problem of having duplicate favourite records.





React

Practise Milestone (Solution)

```
static addToFavourites(homeId) {
  const db = getDb();
  return db
    .collection("favourites")
    .findOne({ homeId: new ObjectId(String(homeId)) })
    .then(existingFavourite => {
      if (!existingFavourite) {
        return db
          .collection("favourites")
          .insertOne({ homeId: new ObjectId(String(homeId)) })
          .then(result => {
            console.log(result);
          });
      }
      return Promise.resolve(); // Do nothing if already exists
    });
}
```





18.5 Define the Logout Feature

1. Define a logout button in nav bar that should come only when user is logged in. Button should be a form that submits to link /logout.
2. Hide the login button in case user is logged in.
3. Handle the logout path and set the isLoggedIn cookie to false.



18.5 Define the Logout Feature

1, 2.

```
<div class="flex space-x-4">
  <% if (!isLoggedIn) { %>
    <a href="/login" class="bg-blue-600 hover:bg-blue-700 text-white p-2 rounded">
      Login
    </a>
  <% } else { %>
    <form action="/logout" method="POST">
      <button type="submit" class="bg-red-600 hover:bg-red-700 text-white p-2 rounded">
        Logout
      </button>
    </form>
  <% } %>
</div>
```



18.5 Define the Logout Feature

3.

```
exports.postLogout = (req, res, next) => {
  res.cookie("isLoggedIn", false);
  res.redirect("/login");
};
```



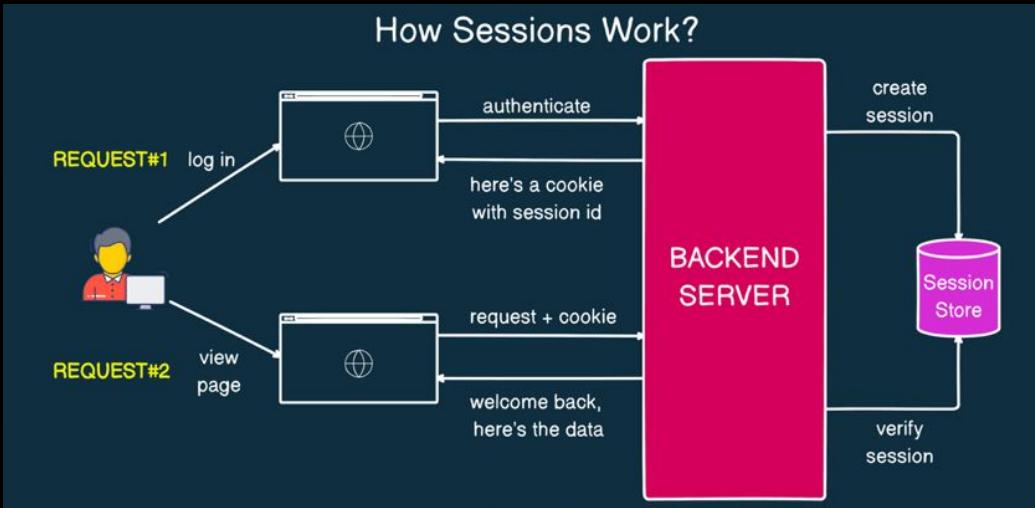
18.6 Problem with *Cookies*

1. Cookies can be intercepted or stolen, posing security risks.
2. They have limited storage capacity (about 4KB).
3. Users can delete or modify cookies, leading to data loss or tampering.
4. Data in cookies is not encrypted, making sensitive information vulnerable.
5. Storing important info in cookies exposes it to client-side attacks.





18.7 What are Sessions



1. Sessions are **server-side storage mechanisms** that track user interactions with a website.
2. They maintain **user state** and data across multiple requests in a web application.
3. Sessions enable **persistent user experiences** by maintaining state between the client and server over **stateless HTTP**.



18.8 Installing Session Package

Pro Teams Pricing Documentation

npm Search packages Sign Up Sign In

express-session DT

1.18.1 • Public • Published a month ago

Readme Code Beta 8 Dependencies 5,041 Dependents 65 Versions

express-session

npm v1.18.1 downloads 7.8M/month ci success coverage 100%

Installation

This is a [Node.js](#) module available through the [npm registry](#). Installation is done using the [npm install command](#):

```
$ npm install express-session
```

API

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

Install

```
> npm i express-session
```

Repository github.com/expressjs/session

Homepage github.com/expressjs/session#readme

Weekly Downloads 1,829,979

Version 1.18.1 License MIT



18.8 Installing Session Package

```
prashantjain@Prashants-MacBook-Pro:~/Desktop/mongoose$ npm install express-session
added 6 packages, and audited 264 packages in 2s
48 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

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

app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
  // Secret key used to sign the session ID cookie and encrypt session data
  secret: 'Complete Coding Secret',
  // Forces session to be saved back to the session store, even if not modified
  resave: false,
  // Forces a session that is "uninitialized" to be saved to the store
  saveUninitialized: true,
}));
```



18.9 Creating a Session

1. Remove setting the cookie and now save the flag in session.
 2. Check the browser for cookie changes.
 3. Log Session in some get request
-
- Sensitive info is stored on server.
 - Same session is valid for all requests from one user.
 - Try to use a different browser and show that session is different
 - Sessions are stored in memory so they reset when server restarts.



18.9 Creating a Sessions

1.

```
exports.postLogin = (req, res, next) => {
  console.log(req.body);
  req.session.isLoggedIn = true;
  res.redirect("/");
};
```

2.

| Application | Filter | | |
|-----------------|-------------|----------------------------|--------|
| | Name | Value | Do... |
| Manifest | connect.sid | s%3Ao1svY1dKn7Pltaijb0U... | loc... |
| Service workers | isLoggedIn | false | loc... |
| Storage | | | |



18.9 Creating a Sessions

3. `exports.getIndex = (req, res, next) => {
 console.log(req.session, req.session.isLoggedIn);
 Home.find().then((registeredHomes) => {`

```
Session {  
  cookie: { path: '/', _expires: null, originalMaxAge: null, httpOnly: true },  
  isLoggedIn: true  
} true
```



18.10 Saving Session in DB

★ 176 **connect-mongodb-session** Lightweight MongoDB-based session store built and maintained by MongoDB.

Heart Pro Teams Pricing Documentation

npm Search packages Sign Up Sign In

connect-mongodb-session DT

5.0.0 • Public • Published 10 months ago

[Readme](#) [Code](#) Beta [2 Dependencies](#) [88 Dependents](#) [41 Versions](#)

connect-mongodb-session

MongoDB-backed session storage for [connect](#) and [Express](#). Meant to be a well-maintained and fully-featured replacement for modules like [connect-mongo](#)

[Build Status](#) coverage 90%

MongoDBStore

This module exports a single function which takes an instance of connect (or Express) and returns a `MongoDBStore` class that can be used to store sessions in MongoDB.

It can store sessions for Express 4

If you pass in an instance of the [express-session module](#) the `MongoDBStore` class will

Install `> npm i connect-mongodb-session`

Repository [github.com/mongodb-js/connect-mon...](#)

Homepage [github.com/mongodb-js/connect-mon...](#)

Weekly Downloads  13,575

Version License



18.10 Saving Session in DB

```
prashantjain@Prashants-MacBook-Pro ~ % npm install connect-mongodb-session
added 3 packages, and audited 267 packages in 2s
48 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

```
const MongoDBStore = require('connect-mongodb-session')(session);

const MONGO_DB_URL = "mongodb+srv://root:root@kgcluster.ie6mb.mongodb.net

const store = new MongoDBStore({
  uri: MONGO_DB_URL,
  collection: 'sessions'
});
```

```
app.use(session({
  // Secret key used to sign the session ID cookie and encrypt session data
  secret: 'Complete Coding Secret',
  // Forces session to be saved back to the session store, even if not modified
  resave: false,
  // Forces a session that is "uninitialized" to be saved to the store
  saveUninitialized: true,
  store: store,
}));
```



Practise Milestone

Take your airbnb forward:

- Cleanup cookie code to use session everywhere.
- Remove the cookie middleware.
- Destroy the session on logout.
- Cleanup Logs.
- Understand why leaving the cookie on logout is fine.





Practise Milestone (Solution)

SEARCH

req.isLoggedIn Aa ab *

req.session.isLoggedIn AB ...

files to include
./5 NodeJs and ExpressJs/13 mongoose

files to exclude

11 results in 5 files - Open in editor

- ✓ JS app.js 5 NodeJs and ExpressJs/13 mo... M 1
if (!req.isLoggedIn || req.session.isLoggedIn) {
- ✓ JS authController.js 5 NodeJs and Expre... U 1
req.isLoggedIn || req.session.isLoggedIn = true;
- ✓ JS errorController.js 5 NodeJs and Expr... M 1
...Page Not Found', isLoggedIn: req.isLoggedIn r...
- ✓ JS hostController.js 5 NodeJs and Expre... M 3
const isLoggedIn = req.isLoggedIn || req.session.i...
isLoggedIn: req.isLoggedIn || req.session.isLogge...
isLoggedIn: req.isLoggedIn || req.session.isLogge...
- ✓ JS storeController.js 5 NodeJs and Expr... M 5
console.log(req.session, req.isLoggedIn || req.ses...
isLoggedIn: req.isLoggedIn || req.session.isLogge...
isLoggedIn: req.isLoggedIn || req.session.isLogge...
isLoggedIn: req.isLoggedIn || req.session.isLogge...
isLoggedIn: req.isLoggedIn || req.session.isLogge...

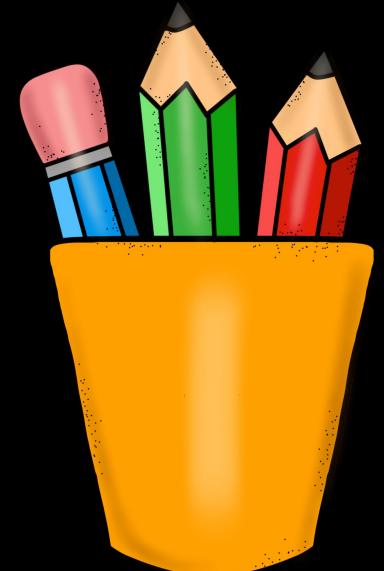
```
exports.postLogout = (req, res, next) => {
  req.session.destroy(() => {
    res.redirect("/login");
  });
};
```

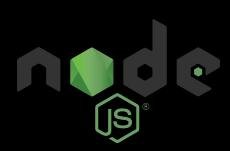




Revision

1. What are Cookies
2. Adding Login Functionality
3. Checking Login State
4. Using a Cookie
5. Define the Logout Feature
6. Problem with Cookies
7. What are Sessions
8. Installing Session Package
9. Creating a Sessions
10. Saving Session in DB







19. Authentication & Authorization

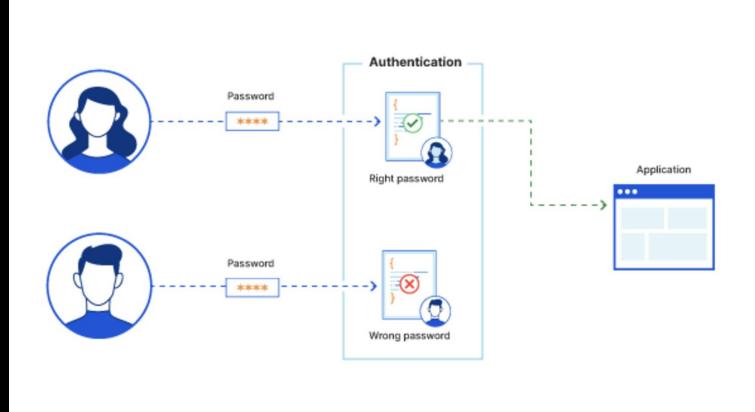
1. What is **Authentication**
2. What is **Authorization**
3. **Authentication vs Authorization**
4. **Session based Authentication**
5. **Signup UI**
6. **Using Express Validator**
7. **Adding User Model**
8. **Encrypting Password**
9. **Implementing Login**





19.1 What is Authentication

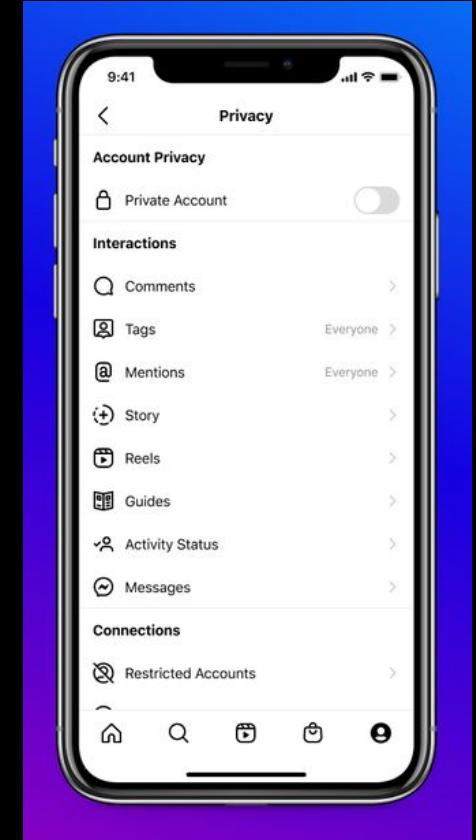
1. Authentication is the process of verifying the identity of a user or system accessing an application.
2. It ensures that only authorized users can access protected resources and features.
3. In ReactJS (Frontend), authentication handles user input for login forms and manages authentication states.
4. In NodeJS (Backend), authentication checks user credentials against a database and issues tokens or sessions.
5. Authentication is crucial for security, protecting data, and providing personalized experiences in web applications.





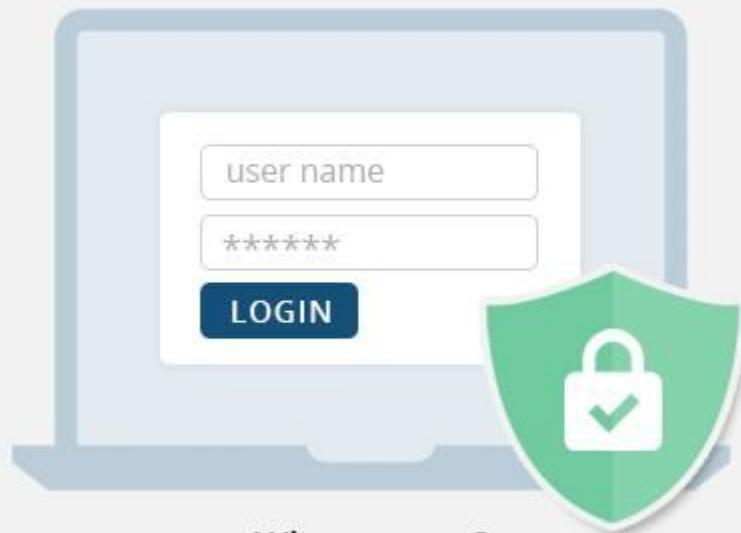
19.2 What is Authorization

1. Authorization is the process of determining what actions a user is permitted to perform within an application.
2. It ensures that users can access only the resources and functionalities they have permission for.
3. In ReactJS (Frontend), authorization controls the display of UI elements and routes based on user roles or permissions.
4. In NodeJS (Backend), authorization involves middleware or logic that checks user permissions before processing requests.
5. Authorization enhances security by restricting access to sensitive data and operations, complementing the authentication process.



19.3 Authentication vs Authorization

Authentication



Who are you?

Validate a system is accessing by the right person

Authorization



Are you allowed to do that?

Check users' permissions to access data



19.3 Authentication vs Authorization

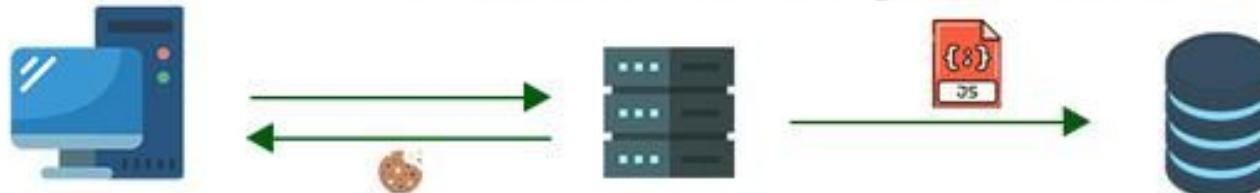
| Aspect | Authentication | Authorization |
|------------------|---|--|
| Definition | Verifies the identity of a user or system | Determines what resources a user can access |
| Purpose | Ensures users are who they claim to be | Grants or denies permissions to resources and actions |
| Process | Validates credentials like usernames and passwords | Checks user privileges and access levels |
| Occurs When | At the start of a session or when accessing secured areas | After authentication, during resource access |
| User Interaction | Requires user input (e.g., logging in) | Usually transparent unless access is denied |
| Managed By | Handled by both frontend and backend systems | Mainly enforced by backend servers |
| Example | User logs into an account with a password | User accesses settings page only if they have admin rights |



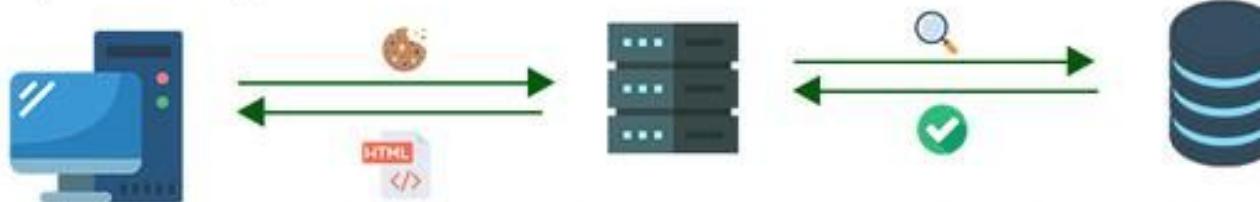
19.4 Session based Authentication

The user sends login request

The server authorizes the login, sends a session to the database, and returns a cookie containing the session ID to the user



The user sends new request
(with a cookie)



The server looks up in the database for the ID Found in the cookie, if the ID is found it sends the requested pages to the user



19.5 Signup UI

1. Define a signup button in navigation bar along with sign-in. It should point to a link /signup.
2. Define a auth/signup.ejs file that has email, password and confirm password fields and submits POST request to /signup
3. Define routes in authRouter and behaviour in authController.
4. Fix the UI of the app to look pretty.



19.5 Signup UI

```
1.  <% if (!isLoggedIn) { %>
    <a href="/signup" class="bg-green-600 hover:bg-green-700 text-white font-semibold py-2.5 px-6
    rounded-lg transition duration-300 ease-in-out transform hover:scale-105 shadow-md">
        | Sign Up
    </a>
    <a href="/login" class="bg-blue-600 hover:bg-blue-700 text-white font-semibold py-2.5 px-6
    rounded-lg transition duration-300 ease-in-out transform hover:scale-105 shadow-md">
        | Login
    </a>
<% } else { %>
```



19.5 Signup UI

2.

```
<%- include('../partials/head') %>
</head>
<body class="min-h-screen bg-gray-100">
  <%- include('../partials/nav') %>
  <h1 class="text-4xl font-bold text-blue-600 mb-8 text-center mt-8">Sign Up Here</h1>
  <div class="flex justify-center">
    <form action="/signup" method="POST" class="w-full max-w-md">

      <input type="email" name="email" placeholder="Enter your email" class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500" required />

      <input type="password" name="password" placeholder="Enter your password" class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500" required />

      <input type="password" name="confirmPassword" placeholder="Confirm your password" class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500" required />

      <div class="flex justify-center">
        <input type="submit" value="Sign Up" class="bg-green-500 hover:bg-green-600 text-white font-semibold py-2 px-4 rounded-lg transition duration-300 ease-in-out transform hover:scale-105 cursor-pointer">
      </div>

    </form>
  </div>
</main>
</body>
</html>
```



19.5 Signup UI

3, 4.

```
authRouter.post("/logout", authController.postLogout);
authRouter.get("/signup", authController.getSignup);
```

```
exports.getSignup = (req, res, next) => {
  res.render("auth/signup", { pageTitle: "Sign Up", isLoggedIn: false });
};
```



19.6 Using Express Validator

1. Add handling for `POST /signup` in auth controller and router.
2. Install `Express Validator`.
3. Use the `email` and `password` validations in the post handler.
4. Change the `signup.ejs` to show the errors. And `accept` the old values.

Version: 7.2.0

express-validator

Overview

express-validator is a set of express.js middlewares that wraps the extensive collection of validators and sanitizers offered by validator.js.

It allows you to combine them in many ways so that you can validate and sanitize your express requests, and offers tools to determine if the request is valid or not, which data was matched according to your validators, and so on.



19.6 Using Express Validator

1.

```
authRouter.get("/signup", authController.getSignup);
authRouter.post("/signup", authController.postSignup);
```

```
exports.postSignup = (req, res, next) => {
  console.log(req.body);
  res.redirect("/login");
};
```

2.

```
prashantjain@Prashants-MacBook-Pro:~/Desktop$ npm install express-validator
added 2 packages, and audited 269 packages in 553ms
48 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```



19.6 Using Express Validator

3.

```
exports.postSignup = [
  // First Name validation
  check('firstName')
    .notEmpty()
    .withMessage('First name is required')
    .trim()
    .isLength({ min: 2 })
    .withMessage('First name must be at least 2 characters long')
    .matches(/^[a-zA-Z\s]+$/)
    .withMessage('First name can only contain letters'),

  // Last Name validation
  check('lastName')
    .notEmpty()
    .withMessage('Last name is required')
    .trim()
    .isLength({ min: 2 })
    .withMessage('Last name must be at least 2 characters long')
    .matches(/^[a-zA-Z\s]+$/)
    .withMessage('Last name can only contain letters'),

  // Email validation
  check('email')
    .isEmail()
    .withMessage('Please enter a valid email')
    .normalizeEmail(),

  // Password validation
  check('password')
    .isLength({ min: 8 })
    .withMessage('Password must be at least 8 characters long')
    .matches(/^[a-z]/)
    .withMessage('Password must contain at least one lowercase letter')
    .matches(/^[A-Z]/)
    .withMessage('Password must contain at least one uppercase letter')
    .matches(/[^@#$%^&*(),.:{}|<>]/)
    .withMessage('Password must contain at least one special character')
    .trim(),

  // Confirm password validation
  check('confirm_password')
    .trim()
    .custom((value, { req }) => {
      if (value !== req.body.password) {
        throw new Error('Passwords do not match');
      }
      return true;
    }),

  // User Type validation
  check('userType')
    .notEmpty()
    .withMessage('User type is required')
    .isIn(['guest', 'host'])
    .withMessage('Invalid user type'),

  // Terms Accepted validation
  check('termsAccepted')
    .notEmpty()
    .withMessage('You must accept the terms and conditions')
    .custom((value) => {
      if (value !== 'on') {
        throw new Error('You must accept the terms and conditions');
      }
      return true;
    })
];
```



19.6 Using Express Validator

3.

```
// Final handler middleware
(req, res, next) => {
  const { firstName, lastName, email, password, userType } = req.body;
  const errors = validationResult(req);

  if (!errors.isEmpty()) {
    return res.status(422).render('auth/signup', {
      pageTitle: 'Sign Up',
      isLoggedIn: false,
      errorMessages: errors.array().map(error => error.msg),
      oldInput: {
        firstName,
        lastName,
        email,
        password,
        userType
      }
    });
  }

  res.redirect('/login');
}
```



19.6 Using Express Validator

```
4. <% if (typeof errorMessages !== 'undefined' && errorMessages && errorMessages.length > 0) { %>
  <div class="■bg-red-100 border ■border-red-400 ■text-red-700 px-4 py-3 rounded relative mb-4"
    role="alert">
    <ul class="list-disc list-inside space-y-1">
      <% errorMessages.forEach(error => { %>
        |   <li><%= error %></li>
        |   <% }); %>
      </ul>
    </div>
<% } %>

<input
  type="text"
  name="firstName"
  placeholder="First Name"
  value="<%= typeof oldInput !== 'undefined' ? oldInput.firstName : '' %>"
  class="w-full px-4 py-2 mb-4 border ■border-gray-300 rounded-lg focus:outline-none
  ■focus:border-blue-500"
/>
```



19.7 Adding User Model

1. Define a new User Model with following fields:
 - a. firstName & lastName (required)
 - b. email (required, unique)
 - c. Password (required)
 - d. userType (possible values 'guest', 'host')
2. In the POST signup handler, create a new user with the fields from request and redirect to login after saving the user.



19.7 Adding User Model

1.

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

const userSchema = new mongoose.Schema({
  firstName: {
    type: String,
    required: true
  },
  lastName: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  },
  userType: {
    type: String,
    required: true,
    enum: ['guest', 'host']
  }
});

module.exports = mongoose.model('User', userSchema);
```



19.7 Adding User Model

```
2. const user = new User({
    firstName: firstName,
    lastName: lastName,
    email: email,
    password: password,
    userType: userType
});

user.save()
  .then(() => {
    res.redirect('/login');
})
  .catch(err => {
    console.log("Error while creating user: ", err);
});
```



19.8 Encrypting Password

1. Understand the problem with plain text passwords.
2. Install a package named: bcryptjs
3. Hash the password before saving password.
4. Understand that even server does not have the password.



19.8 Encrypting Password

2. prashantjain@Prashants-MacBook-Pro 13 mongoose % npm install bcryptjs
added 1 package, and audited 270 packages in 513ms
48 packages are looking for funding
 run `npm fund` for details
found 0 vulnerabilities



19.8 Encrypting Password

```
3. bcrypt.hash(password, 12).then(hashedPassword => {
    const user = new User({
        firstName: firstName,
        lastName: lastName,
        email: email,
        password: hashedPassword,
        userType: userType
    });

    user.save()
        .then(() => {
            res.redirect('/login');
        })
        .catch(err => {
            console.log("Error while creating user: ", err);
        });
});
```



19.9 Implementing Login

1. Read the **email** and **password** from the request body and find the user with the **email** from the **Users** collection.
2. If the **user is not found** send an error and re-render the login page, also make changes to the **login page** to show errors.
3. If the **user is found**, then use the **bcrypt compare function** to **match** the entered password, If password does not match **send another error**, otherwise **create a login session** and **redirect user to the home**.



19.9 Implementing Login

1.

```
exports.postLogin = async (req, res, next) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ email });

    if (!user) {
      return res.render('auth/login', {
        pageTitle: 'Login',
        isLoggedIn: false,
        errorMessage: 'Invalid Email'
      });
    }

    res.redirect("/");
  } catch (err) {
    console.log("Error while logging in: ", err);
  }
};
```



19.9 Implementing Login

2.

```
<% if (typeof errorMessage !== 'undefined' && errorMessage) { %>
  <div class="■bg-red-100 border ■border-red-400 ■text-red-700 px-4 py-3 rounded relative mb-4"
       role="alert">
    <span class="block sm:inline"><%= errorMessage %></span>
  </div>
<% } %>
```



19.9 Implementing Login

3.

```
const isMatch = await bcrypt.compare(password, user.password);

if (!isMatch) {
  return res.render('auth/login', {
    pageTitle: 'Login',
    isLoggedIn: false,
    errorMessage: 'Invalid Password'
  });
}

req.session.isLoggedIn = true;
req.session.user = user;
await req.session.save();

res.redirect("/");
} catch (err) {
  console.log("Error while logging in: ", err);
}
```



19.10 Adding User Functions

1. Make the navigation bar items display only on the basis of userType by passing the user object to all views.
2. Add a field in User Model names favouriteHomes, which is an array of home ids.
3. Remove the Favourite Model and the Pre hook from the Home Model.
4. Make the favourite user specific and change the following methods:
 - a. postAddFavourites
 - b. getFavourites
 - c. postRemoveFavourite



19.10 Adding User Functions

1.

```
<% if (isLoggedIn) { %>
<% if (user.userType === 'guest') { %>
  <a href="/homes" class="■bg-blue-600 ■hover:bg-blue-700 ■text-white
    font-semibold py-2.5 px-6 rounded-lg transition duration-300 ease-in-out
    transform hover:scale-105 shadow-md">
    | Homes
  </a>
  <a href="/favourites" class="■bg-blue-600 ■hover:bg-blue-700 ■text-white
    font-semibold py-2.5 px-6 rounded-lg transition duration-300 ease-in-out
    transform hover:scale-105 shadow-md">
    | Favourites
  </a>
<% } %>
<% if (user.userType === 'host') { %>
  <a href="/host/host-homes" class="■bg-blue-600 ■hover:bg-blue-700 ■text-wh
    font-semibold py-2.5 px-6 rounded-lg transition duration-300 ease-in-out
    transform hover:scale-105 shadow-md">
    | Host Homes
  </a>
  <a href="/host/add-home" class="■bg-blue-600 ■hover:bg-blue-700 ■text-whit
    font-semibold py-2.5 px-6 rounded-lg transition duration-300 ease-in-out
    transform hover:scale-105 shadow-md">
    | Add Home
  </a>
<% } %>
```

isLoggedIn: req.session.isLoggedIn,
user: req.session.user,



19.10 Adding User Functions

```
2. userType: {  
    type: String,  
    required: true,  
    enum: ['guest', 'host']  
},  
favouriteHomes: [{  
    type: mongoose.Schema.Types.ObjectId,  
    ref: 'Home'  
}]
```



19.10 Adding User Functions

4.a.

```
exports.postAddFavourites = (req, res, next) => {
  const homeId = req.body.id;
  const userId = req.session.user._id;

  User.findById(userId)
    .then(user => {
      if (!user.favouriteHomes.includes(homeId)) {
        user.favouriteHomes.push(homeId);
        return user.save();
      }
      return user;
    })
    .then(() => {
      res.redirect("/favourites");
    })
    .catch((err) => {
      console.log("Error while adding to favourites", err);
      res.redirect("/favourites");
    });
};
```



19.10 Adding User Functions

```
4.b. exports.getFavourites = (req, res, next) => {
    const userId = req.session.user._id;
    User.findById(userId)
        .populate('favouriteHomes')
        .then(user) => {
            res.render("store/favourites", {
                homes: user.favouriteHomes,
                pageTitle: "Favourites",
                isLoggedIn: req.session.isLoggedIn,
                user: req.session.user,
            });
        });
};
```



19.10 Adding User Functions

4.c.

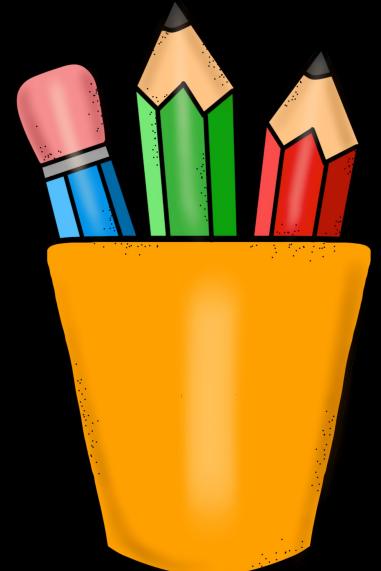
```
exports.postRemoveFavourite = (req, res, next) => {
  const homeId = req.params.homeId;
  const userId = req.session.user._id;

  User.findById(userId)
    .then(user => {
      user.favouriteHomes = user.favouriteHomes.filter(id => id.toString()
        !== homeId);
      return user.save();
    })
    .then(() => {
      res.redirect("/favourites");
    })
    .catch(error => {
      console.log("Error while remove from favourites ", error);
      res.redirect("/favourites");
    });
};
```



Revision

1. What is Authentication
2. What is Authorization
3. Session based Authentication
4. Authentication vs Authorization
5. Signup UI
6. Using Express Validator
7. Adding User Model
8. Encrypting Password
9. Implementing Login
10. Adding User Functions





Practise Milestone

Take your airbnb forward:

1. Add a field to **Home model** for a host user id.
2. Change the home creation **to pass the current host id.**
3. Change the **UI of /host-homes** to only use homes belonging **to the particular host.**





Practise Milestone (Solution)

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

const homeSchema = new mongoose.Schema({
  houseName : {type: String, required: true},
  price: {type: Number, required: true},
  location: {type: String, required: true},
  rating: {type: Number, required: true},
  photoUrl: String,
  description: String,
  hostId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  }
});

module.exports = mongoose.model("Home", homeSchema);
```





Practise Milestone (Solution)

2.

```
const newHome = new Home({
    houseName,
    price,
    location,
    rating,
    photoUrl,
    description,
    hostId: req.session.user._id
});
```





Practise Milestone (Solution)

3.

```
exports.getHostHomes = (req, res, next) => {
  const userId = req.session.user._id;
  Home.find({ hostId: userId }).then((registeredHomes) => {
    console.log(registeredHomes);
    res.render("host/host-homes", {
      homes: registeredHomes,
      pageTitle: "Host Homes",
      isLoggedIn: req.session.isLoggedIn,
      user: req.session.user,
    });
  });
};
```





20. Emails & Advanced Auth

1. Why to send Email
2. Using SendGrid
3. Sending Email
4. Forgot Password Wireframe
5. Sending OTP
6. Reset Password





20.1 Why to send Email

1. **Welcome Emails:** Send a confirmation or greeting when a user successfully signs up.
2. **Password Reset Emails:** Provide secure links or OTPs for users to reset their passwords.
3. **Order Confirmation:** Send receipts or confirmations after a purchase is completed.
4. **Account Activity Alerts:** Notify users of important events like login from a new device.
5. **Newsletter and Promotions:** Distribute newsletters or promotional offers to subscribed users.





20.2 Using SendGrid

**twilio
SendGrid**

Products Why SendGrid Resources Developers Pricing Contact us Sign in Start for free

Twilio SendGrid

Everything you need to deliver emails at scale

- 148+ billion emails sent every month
- 82,000+ customers
- 100+ full-time email deliverability experts

[Start for free](#) [See plans & pricing →](#)

No credit card required | Get started quickly | Access to all Twilio products

Email API and SMTP service for developers
Trusted API for email delivery at scale.
[Learn more →](#)

Email campaigns for marketers
Tools to create engaging email campaigns.
[Learn more →](#)

Uber **Spotify** **airbnb** **yelp** **glassdoor** **instacart**



20.2 Using SendGrid

twilio
sendgrid

Products Why SendGrid Resources Developers Pricing Contact us Sign in Start for free

SendGrid pricing

Start for free. Then pay as you go.

- Sign up for a free trial - no credit card required
- Only pay for the volume and features you need
- Unlock volume discounts as you scale

[Start for free](#)



Email API
Integrate email into your app or website.

Marketing Campaigns
Design and send email marketing campaigns and automations.

Email API Plans

Pricing is determined by monthly email volume and features. Select your email volume below to see which plans are right for you.

How many emails would you like to send per month?



</>

| Volume | Plan | Features |
|------------|------------|--|
| <3,000 | Free | Up to 3,000 emails/month, 100 API requests |
| 50,000 | Starter | Up to 50,000 emails/month, 1,000 API requests |
| 100,000 | Standard | Up to 100,000 emails/month, 2,000 API requests |
| 300,000 | Pro | Up to 300,000 emails/month, 5,000 API requests |
| 700,000 | Business | Up to 700,000 emails/month, 10,000 API requests |
| 1,500,000 | Enterprise | Up to 1,500,000 emails/month, 20,000 API requests |
| 2,500,000 | Custom | Up to 2,500,000 emails/month, 50,000 API requests |
| 5,000,000+ | Custom | Up to 5,000,000+ emails/month, 100,000+ API requests |



20.2 Using SendGrid



API Keys

[Create API Key](#)

Get started creating API Keys

API keys help protect the sensitive areas of your SendGrid account (e.g. contacts and account settings). To control and limit access of API users, you can create multiple API keys, each with different permissions.



20.2 Using SendGrid



API Key Created

Please copy this key and save it somewhere safe.

For security reasons, we cannot show it to you again

Copied!

```
SG.0BsQdZJqSzmcAofUQqeZBQ.z-DbZaCPWs-zkAd3UfzS_YA5fqYH8UUUXg2FANf7mOyl
```

Done



20.2 Using SendGrid

Welcome

▼ How to start sending mail

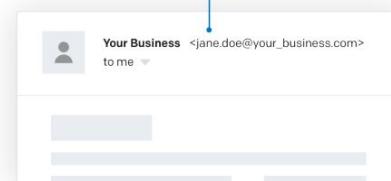
1 Send a single send

Your sender identity is the “from” email address your recipients see in their inbox. Once done, set up your integration for access to robust features.

[Create sender identity →](#)



[Learn more about sender identity](#)



2 Learn about email sending options

Complete the first step to unlock email sending options.



20.2 Using SendGrid

Sender Authentication /

Single Sender Verification (i)

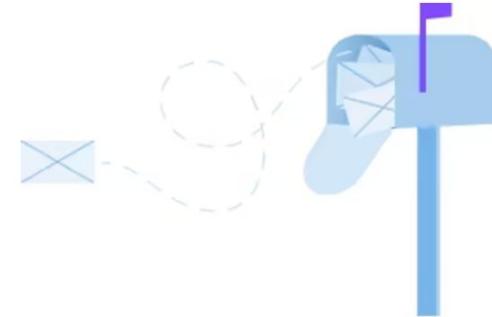
SENDERS

ADDRESS

Complete Coding

FROM contact@completecoding.in
REPLY contact@completecoding.in

Plot No 15
Sector 10A, Vasundha
Ghaziabad, 201012 IN



Sender has been created

To verify sender identity, check your inbox
at contact@completecoding.in.

Resend email

Close



20.2 Using SendGrid

Pro Teams Pricing Documentation

npm



Search packages

Search

Sign Up

Sign In

@sendgrid/mail ts

8.1.4 • Public • Published a month ago

Readme

Code

Beta

2 Dependencies

1,320 Dependents

52 Versions

build passing npm package 8.1.4

This package is part of a monorepo, please see [this README](#) for details.

Mail Service for the SendGrid v3 Web API

This is a dedicated service for interaction with the mail endpoint of the [SendGrid v3 API](#).

Installation

Install

`> npm i @sendgrid/mail`



Repository

github.com/sendgrid/sendgrid-nodejs

Homepage

sendgrid.com



20.2 Using SendGrid

```
prashantjain@Prashants-MacBook-Pro 13 mongoose % npm install @sendgrid/mail  
added 11 packages, and audited 281 packages in 3s  
  
49 packages are looking for funding  
  run `npm fund` for details  
  
1 high severity vulnerability  
  
To address all issues, run:  
  npm audit fix  
  
Run `npm audit` for details.
```



20.3 Sending Email

```
const sgMail = require('@sendgrid/mail');
const SENDGRID_API_KEY = 'SG.0BsQdZJqSzmcAofUQqeZBQ.
z-DbZaCPWs-zkAd3UfzS_YA5fqYH8UUUXg2FANf7m0yI';

bcrypt.hash(password, 12)
  .then(hashedPassword => {
    const user = new User({
      firstName: firstName,
      lastName: lastName,
      email: email,
      password: hashedPassword,
      userType: userType
    });
    return user.save();
  })
  .then(() => {
    // Send welcome email
    sgMail.setApiKey(SENDGRID_API_KEY);
    const msg = {
      to: email,
      from: 'contact@completecoding.in', // Change to your verified sender
      subject: 'Welcome to Complete Coding!!',
      html: `<h1>Welcome ${firstName}!</h1><p>Welcome to our family!We're excited to
      have you on board.</p><p>Best regards,<br>The Team</p>`
    };
    return sgMail.send(msg);
  })
  .then(() => {
    res.redirect('/login');
  })
  .catch(error => {
    console.error(error);
  })
  .finally(() => {
    res.end();
  })
}
```



20.3 Sending Email

Welcome to Complete Coding! ➤ Inbox ×



contact@completecoding.in via sendgrid.net
to me ▾

Welcome prashant!

Welcome to our family! We're excited to have you on board.

Best regards,
The Team



20.4 Forgot Password Wireframe

1. Define a new `forgot.ejs` file using the `login.ejs` file with `email` field and a `button` to Reset Password.
2. Add a link to login page for forgot password to `/forgot-password`
3. Add entry in the `auth router and controller` to show the page.
4. Submitting the form should go to `POST /forgot-password` which will find the user by `email-id` and redirect him to a new page `/reset-password` passing `email-id` as a parameter.



20.4 Forgot Password Wireframe

1.

```
<form action="/forgot-password" method="POST" class="w-full max-w-md">

  <% if (typeof errorMessage !== 'undefined' && errorMessage) { %>
    <div class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded relat
      mb-4" role="alert">
      <span class="block sm:inline"><%= errorMessage %></span>
    </div>
  <% } %>

  <input
    type="email"
    name="email"
    value="<%= typeof oldEmail !== 'undefined' ? oldEmail : '' %>"
    placeholder="Enter your email"
    class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none
      focus:border-blue-500"
  />

  <div class="flex justify-center">
    <input
      type="submit"
      value="Reset Password"
      class="bg-blue-500 hover:bg-blue-600 text-white font-semibold py-2 px-4 rounded
        transition duration-300 ease-in-out transform hover:scale-105 cursor-pointer"
    >
  </div>
</form>
```



20.4 Forgot Password Wireframe

2.

```
<div class="flex justify-between items-center mb-4">
  <a href="/forgot-password" class="text-blue-500 hover:text-blue-600
  text-sm">Forgot Password?</a>
</div>
```

3.

```
exports.getForgotPassword = (req, res, next) => {
  res.render("auth/forgot", { pageTitle: "Forgot Password", isLoggedIn: false,
  user: req.session.user });
};
```

```
authRouter.get("/forgot-password", authController.getForgotPassword);
authRouter.post("/forgot-password", authController.postForgotPassword);
```



20.4 Forgot Password Wireframe

4.

```
exports.postForgotPassword = async (req, res, next) => {
  const { email } = req.body;
  try {
    const user = await User.findOne({ email: email });
    if (!user) {
      return res.render('auth/forgot', {
        pageTitle: 'Forgot Password',
        isLoggedIn: false,
        user: req.session.user,
        errorMessage: 'No account found with this email'
      });
    }
    res.redirect(`/reset-password?email=${email}`);
  } catch (err) {
    console.log('Error in forgot password:', err);
    return res.render('auth/forgot', {
      pageTitle: 'Forgot Password',
      isLoggedIn: false,
      user: req.session.user,
      errorMessage: err.message
    });
  }
};
```



20.5 Sending OTP

1. Add **OTP** and **OTP-Expiry** fields to User model.
2. Generate a random 6 digit **OTP**, to be sent in the password reset email.
3. In the **user** object add value of both fields, **save the user**.
4. Send the same **OTP** to the user in an **email** along with a link to **/reset-password** page.



20.5 Sending OTP

```
1. favouriteHomes: [{}  
  |   type: mongoose.Schema.Types.ObjectId,  
  |   ref: 'Home'  
  |],  
  otp: {}  
  |   type: String,  
  |   required: false  
  |},  
  otpExpiry: {}  
  |   type: Date,  
  |   required: false  
  |}
```



20.5 Sending OTP

```
exports.postForgotPassword = async (req, res, next) => {
  const { email } = req.body;
  try {
    const user = await User.findOne({ email });
    if (!user) {
      throw new Error('No account found with this email');
    }

    const otp = Math.floor(100000 + Math.random() * 900000).toString();
    user.otp = otp;
    user.otpExpiry = Date.now() + 5 * 60 * 1000;
    await user.save();

    sgMail.setApiKey(SENDGRID_API_KEY);
    const msg = {
      to: email,
      from: 'contact@completecoding.in',
      subject: 'Reset Password OTP',
      html: `<h1>Reset Password OTP</h1><p>Your OTP is ${otp}. It is valid for 5 minutes.</p>`;
    };
    await sgMail.send(msg);

    res.redirect(`/reset-password?email=${email}`);
  } catch (err) {
    console.log('Error in forgot password:', err);
    return res.render('auth/forgot', {
      pageTitle: 'Forgot Password',
      isLoggedIn: false,
      user: req.session.user,
      oldEmail: email,
      errorMessage: err.message
    });
  }
};
```

2,3.

4.



20.6 Reset Password

1. Define a new `reset-password.ejs` file using the `signup.ejs` file that shows `email`, `otp`, `password`, `confirm_password` and a button to Reset Password.
2. Define a GET `/reset-password` route and a controller entry in auth files
3. Submitting the form should go to POST `/reset-password` which will log the request body.
4. Add same validations on `password` and `confirm_password`.
5. In the POST controller, take out the email-id and find the user. Check if user is found and the expiration date is greater than now, and the OTP matches, reset the password. And redirect to login page.



20.6 Reset Password

```
1. <input type="email" name="email" placeholder="Enter your Email" value="<%=\n  !== 'undefined' ? email : '' %>" readonly class="w-full px-4 py-2 mb-4 border\n  border-gray-300 rounded-lg bg-gray-100"/>\n\n<input type="text" name="otp" placeholder="Enter OTP" class="w-full px-4 py-2 mb-4 border\n  border-gray-300 rounded-lg focus:outline-none border-blue-500"/>\n\n<input type="password" name="password" placeholder="New Password" class="w-full px-4 py-2\n  mb-4 border border-gray-300 rounded-lg focus:outline-none border-blue-500"/>\n\n<input type="password" name="confirm_password" placeholder="Confirm New Password"\n  class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none\n  border-blue-500"/>\n\n<div class="flex justify-center">\n  <input type="submit" value="Update Password" class="bg-blue-500 hover:bg-blue-600\n    text-white font-semibold py-2 px-4 rounded-lg transition duration-300 ease-in-out\n    transform hover:scale-105 cursor-pointer">\n</div>
```



20.6 Reset Password

2.

```
authRouter.get("/reset-password", authController.getResetPassword);
authRouter.post("/reset-password", authController.postResetPassword);
```

```
exports.getResetPassword = (req, res, next) => {
  const { email } = req.query;
  res.render("auth/reset_password", { pageTitle: "Reset Password", isLoggedIn: false,
    user: req.session.user, email: email });
};
```

3.

```
exports.postResetPassword = async (req, res, next) => {
  const { email, otp, password } = req.body;
  console.log(email, otp, password);
  return res.redirect('/login');
};
```



20.6 Reset Password

4.

```
exports.postResetPassword = [
  // Password validation
  check('password')
    .isLength({ min: 8 })
    .withMessage('Password must be at least 8 characters long')
    .matches(/^[a-z]*/
    .withMessage('Password must contain at least one lowercase letter')
    .matches(/^[A-Z]*/
    .withMessage('Password must contain at least one uppercase letter')
    .matches(/[^@#$%^&*(),.?":{}|<>]*/
    .withMessage('Password must contain at least one special character')
    .trim(),

  // Confirm password validation
  check('confirm_password')
    .trim()
    .custom((value, { req }) => {
      if (value !== req.body.password) {
        throw new Error('Passwords do not match');
      }
      return true;
    }),
  async (req, res, next) => {
```



20.6 Reset Password

5.

```
async (req, res, next) => {
  const { email, otp, password } = req.body;
  try {
    const user = await User.findOne({ email: email });

    if (!user) {
      throw new Error('No user found with this email');
    } else if (user.otpExpiry < Date.now()) {
      throw new Error('OTP has expired. Please request a new one.');
    } else if (user.otp !== otp) {
      throw new Error('Invalid OTP');
    }

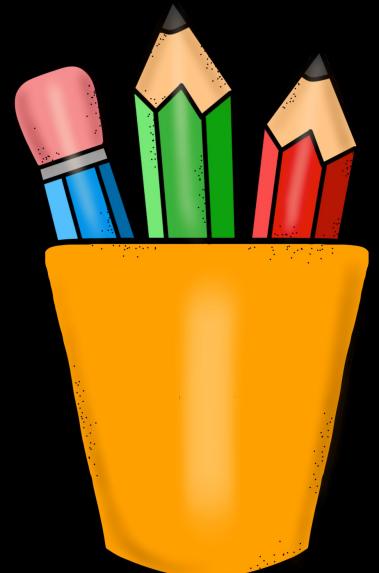
    const hashedPassword = await bcrypt.hash(password, 12);
    user.password = hashedPassword;
    user.otp = undefined;
    user.otpExpiry = undefined;
    await user.save();

    return res.redirect('/login');
  } catch (err) {
    console.log('Error in reset password:', err);
    return res.render('auth/reset_password', {
      pageTitle: 'Reset Password',
      isLoggedIn: false,
      user: req.session.user,
      email: email,
      errorMessage: err.message
    });
  }
}
```



Revision

1. Why to send Email
2. Using SendGrid
3. Sending Email
4. Forgot Password Wireframe
5. Sending OTP
6. Reset Password





Practise Milestone

Take your airbnb forward:

1. Takeout the validations in a separate file.
2. Define a partial for showing errors and add it to all files where it is used.





Practise Milestone (Solution)

1.

```
const { check } = require('express-validator');

exports.firstNameValidator = check('firstName')
    .notEmpty()
    .withMessage("First name is mandatory")
    .trim()
    .isLength({min: 2})
    .withMessage('First Name should be minium 2 chars')
    .matches(/^[a-zA-Z\s]+$/)
    .withMessage('First Name should only contain english aplhabets');

exports.lastNameValidator = check('lastName')
    .trim()
    .matches(/^[a-zA-Z\s]*$/)
    .withMessage('Last Name should only contain english aplhabets');

exports.emailValidator = check('email')
    .isEmail()
    .withMessage('Please enter a valid email')
    .normalizeEmail();

exports.passwordValidator = check('password')
    .trim()
```





Practise Milestone (Solution)

1.

```
const {
  firstNameValidator,
  lastNameValidator,
  emailValidator,
  passwordValidator,
  confirmPasswordValidator,
  userTypeValidator,
  termsAcceptedValidator
} = require('./validations');

exports.postSignup = [
  firstNameValidator,
  lastNameValidator,
  emailValidator,
  passwordValidator,
  confirmPasswordValidator,
  userTypeValidator,
  termsAcceptedValidator,

  // Final handler middleware
  (req, res, next) => {
    const { firstName, lastName, email, password, userType } = req.body;
    const errors = validationResult(req);
```

```
  exports.postResetPassword = [
    passwordValidator,
    confirmPasswordValidator,

    async (req, res, next) => {
      const { email, otp, password } = req.body;
      try {
```



Practise Milestone (Solution)

2.

```
<% if (typeof errorMessages !== 'undefined' && errorMessages && errorMessages.length > 0) { %>
  <ul class="■bg-red-100 border ■border-red-400 ■text-red-700 px-4 py-3 rounded mb-4">
    <% errorMessages.forEach(errorMessage => { %>
      <li class="flex items-center">
        <svg class="w-4 h-4 mr-2" fill="currentColor" viewBox="0 0 20 20">
          <path fill-rule="evenodd" d="M10 18a8 8 0 100-16 8 8 0 00 16zM8.707 7.293a1 1 0 00-1.414 1.414L8.586 10l-1.293 1.293a1 1 0 101.414 1.414L10 11.414L1.293 1.293a1 1 0 001.414-1.414L11.414 10l1.293-1.293a1 1 0 00-1.414-1.414L10 8.586 8.707 7.293z" clip-rule="evenodd"/>
        </svg>
        <%= errorMessage %>
      </li>
    <% }) %>
  </ul>
<% } %>
```

```
<form action="/signup" method="POST" class="w-full max-w-md">

  <%- include('../partials/error-messages') %>

  <input
    type="text"
    name="firstName"
```





22.8 Using Hosting Provider

Learn | Discover | Product documentation | Development languages | Topics

Search | Sign in

Azure Products | Architecture | Develop | Learn Azure | Troubleshooting | Resources

Portal | Free account

Filter by title

Learn / Azure / Azure CLI /

How to install the Azure CLI

Article • 10/16/2024 • 19 contributors

Feedback

In this article

- Install
- FAQ
- See also

The Azure CLI is available to install in Windows, macOS and Linux environments. It can also be run in a Docker container and Azure Cloud Shell.

Install

The current version of the Azure CLI is 2.67.0. For information about the latest release, see the [release notes](#). To find your installed version and see if you need to update, run `az version`.

- Install on Windows
- Install on macOS

Additional resources

Training

Module [Control Azure services with the CLI - Training](#)
Learn the steps to install the Azure CLI locally, create a website, and manage Azure resources using the CLI.

Certification [Microsoft Certified: Azure Fundamentals - Certifications](#)
Demonstrate foundational knowledge of cloud concepts, core Azure services, plus Azure management and governance features and tools.

Documentation

Get started with Azure Command-Line Interface (CLI)
Learn how to start using the Azure CLI by completing common commands. You can begin using the Azure CLI by running it in an Azure Cloud Shell environment.

Install the Azure CLI for Windows
To install the Azure CLI on Windows, you must use PowerShell, or an MSI installer, which gives you access to the CLI through the Windows Command Prompt (CMD).



22.8 Using Hosting Provider

Install or update

The MSI and ZIP distributable are used for installing or updating the Azure CLI on Windows. You don't need to uninstall current versions before using the MSI installer because the MSI updates any existing version.

ⓘ Important

After the installation is complete, you will need to close and reopen any active terminal window to use the Azure CLI.

[Microsoft Installer \(MSI\)](#)

[Microsoft Installer \(MSI\) with PowerShell](#)

[Windows Package Manager](#)

[ZIP Package](#)

Latest version

Download and install the latest release of the Azure CLI. When the installer asks if it can make changes to your computer, select the "Yes" box.

[Latest MSI of the Azure CLI \(32-bit\)](#)

[Latest MSI of the Azure CLI \(64-bit\)](#)

If you have previously installed the Azure CLI, running either the 32-bit or 64-bit MSI will overwrite an existing installation.



22.8 Using Hosting Provider





22.8 Using Hosting Provider

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

PS C:\Users\Prashant> az --version

azure-cli 2.67.0

core 2.67.0

telemetry 1.1.0

Dependencies:

msal 1.31.0

azure-mgmt-resource 23.1.1

Python location 'C:\Program Files\Microsoft SDKs\Azure\CLI2\python.exe'

Extensions directory 'C:\Users\Prashant\.azure\cliextensions'

Python (Windows) 3.12.7 (tags/v3.12.7:0b05ead, Oct 1 2024, 03:06:41) [MSC v.1941 64 bit (AMD64)]

Legal docs and information: aka.ms/AzureCliLegal

Your CLI is up-to-date.



22.8 Using Hosting Provider

```
PS C:\Users\Prashant> az login
Select the account you want to log in with. For more information on login with Azure CLI, see https://go.microsoft.com/fwlink/?linkid=2271136
Retrieving tenants and subscriptions for the selection...
[Tenant and subscription selection]
No   Subscription name   Subscription ID           Tenant
---   -----
[1] *  Free Trial        166737a1-e433-403b-aacb-e39a9d24b588  Default Directory

The default is marked with an *; the default tenant is 'Default Directory' and subscription is 'Free Trial' (166737a1-e433-403b-aacb-e39a9d24b588).
Select a subscription and tenant (Type a number or Enter for no changes): 1

Tenant: Default Directory
Subscription: Free Trial (166737a1-e433-403b-aacb-e39a9d24b588)

[Announcements]
With the new Azure CLI login experience, you can select the subscription you want to use more easily. Learn more about it and its configuration at https://go.microsoft.com/fwlink/?linkid=2271236

If you encounter any problem, please open an issue at https://aka.ms/azclibug

[Warning] The login output has been updated. Please be aware that it no longer displays the full list of available subscriptions by default.
```



22.8 Using Hosting Provider

Deploy to Azure

Before you continue, ensure that you have all the prerequisites installed and configured.

Note

For your Node.js application to run in Azure, it needs to listen on the port provided by the `PORT` environment variable. In your generated Express app, this environment variable is already used in the startup script `bin/www`. (Search for `process.env.PORT`.)

In the terminal, ensure you're in the `myExpressApp` directory, and deploy the code in your local folder (`myExpressApp`) using the `az webapp up` command:

Deploy to Linux

Deploy to Windows

Azure CLI

Copy

```
az webapp up --sku F1 --name <app-name>
```



22.8 Using Hosting Provider

```
PS C:\Users\Prashant\Documents\workspaces\MERNLiveProject> az webapp up --sku F1 --name mern-live-backend
The webapp 'mern-live-backend' doesn't exist
Creating Resource group 'omprashantjain_rg_7471' ...
Resource group creation complete
Creating AppServicePlan 'omprashantjain_asp_8475' or Updating if already exists
 Readonly attribute name will be ignored in class <class 'azure.mgmt.web.v2023_01_01.models._models_py3.AppServicePlan'>
Creating webapp 'mern-live-backend' ...
Configuring default logging for the app, if not already enabled
Creating zip with contents of dir C:\Users\Prashant\Documents\workspaces\MERNLiveProject ...
Getting scm site credentials for zip deployment
Starting zip deployment. This operation can take a while to complete ...
Deployment endpoint responded with status code 202
Polling the status of async deployment. Start Time: 2024-12-01 07:01:15.254002+00:00 UTC
Status: Building the app... Time: 1(s)
Status: Building the app... Time: 18(s)
Status: Building the app... Time: 35(s)
Status: Build successful. Time: 51(s)
Status: Starting the site... Time: 67(s)
Status: Starting the site... Time: 83(s)
Status: Starting the site... Time: 99(s)
Status: Starting the site... Time: 115(s)
Status: Starting the site... Time: 672(s)
Status: Site failed to start. Time: 689(s)
Deployment failed because the site failed to start within 10 mins.
InprogressInstances: 0, SuccessfulInstances: 0, FailedInstances: 1
Error: Deployment for site 'mern-live-backend' with DeploymentId '740108ab-de7d-4235-a240-430f0d16013c' failed because the worker process
start within the allotted time.
```



22.8 Using Hosting Provider

Update appName, .gitignore, and create env files

```
{} package.json
```

```
↳ .gitignore X
```

```
↳ .gitignore
```

```
1 @output.css
```

```
{} package.json > {} devDependencies
1 {
2   "name": "mern-live-backend",
3   "version": "1.0.0",
4   "main": "app.js",
```

```
$ .env.development
```

```
$ .env.example
```

```
$ .env.production
```

```
↳ .gitignore
```

```
$ .env.production
```

```
1 MONGO_DB_USERNAME=root
2 MONGO_DB_PASSWORD=root
3 MONGO_DB_DATABASE=airbnb
4
5 FROM_EMAIL=contact@completecoding.in
6 SENDGRID_API_KEY=SG.gcT1EgxbSFWI8Lu_jHo8rQ.JkyF
```



22.8 Using Hosting Provider

ⓘ Note

The `az webapp up` command does the following actions:

- Create a default resource group.
- Create a default App Service plan.
- Create an app with the specified name.
- Zip deploy all files from the current working directory, with build automation enabled.
- Cache the parameters locally in the `.azure/config` file so that you don't need to specify them again when deploying later with `az webapp up` or other `az webapp` commands from the project folder. The cached values are used automatically by default.



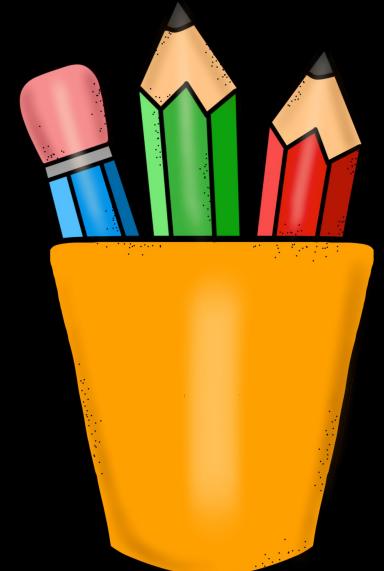
22.8 Using Hosting Provider

```
PS C:\Users\Prashant\Documents\workspaces\MERNLiveProject> az webapp up --name mern-live-backend
Webapp 'mern-live-backend' already exists. The command will deploy contents to the existing app.
Creating AppServicePlan 'omprashantjain_asp_8475' or Updating if already exists
Readonly attribute name will be ignored in class <class 'azure.mgmt.web.v2023_01_01.models._models_py3.AppServicePlan'>
Creating zip with contents of dir C:\Users\Prashant\Documents\workspaces\MERNLiveProject ...
Getting scm site credentials for zip deployment
Starting zip deployment. This operation can take a while to complete ...
Deployment endpoint responded with status code 202
Status: Building the app... Time: 1(s)
Status: Building the app... Time: 21(s)
Status: Build successful. Time: 37(s)
Status: Starting the site... Time: 53(s)
Status: Starting the site... Time: 69(s)
Status: Starting the site... Time: 86(s)
Status: Site started successfully. Time: 102(s)
You can launch the app at http://mern-live-backend.azurewebsites.net
Setting 'az webapp up' default arguments for current directory. Manage defaults with 'az configure --scope local'
--resource-group/-g default: omprashantjain_rg_7471
--sku default: F1
--plan/-p default: omprashantjain_asp_8475
--location/-l default: canadacentral
--name/-n default: mern-live-backend
{
  "URL": "http://mern-live-backend.azurewebsites.net",
  "appserviceplan": "omprashantjain_asp_8475",
  "location": "canadacentral",
  "name": "mern-live-backend",
  "os": "Linux",
  "resourcegroup": "omprashantjain_rg_7471",
  "runtime_version": "node|16-LTS",
  "runtime_version_detected": "0.0",
  "sku": "FREE",
  "src_path": "C:\\\\Users\\\\Prashant\\\\Documents\\\\workspaces\\\\MERNLiveProject"
}
```



Revision

1. What is Deployment?
2. Cloud vs Local Deployments
3. Using Environment Variables
4. Secure Response Headers
5. Use Compressed Assets
6. Request-Response Logging
7. Using SSL/TLS Encryption
8. Using Hosting Provider



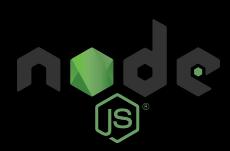


Practise Milestone

Take your **airbnb** forward:

1. Check the UI of Azure Cloud Services and familiarize yourself with it.







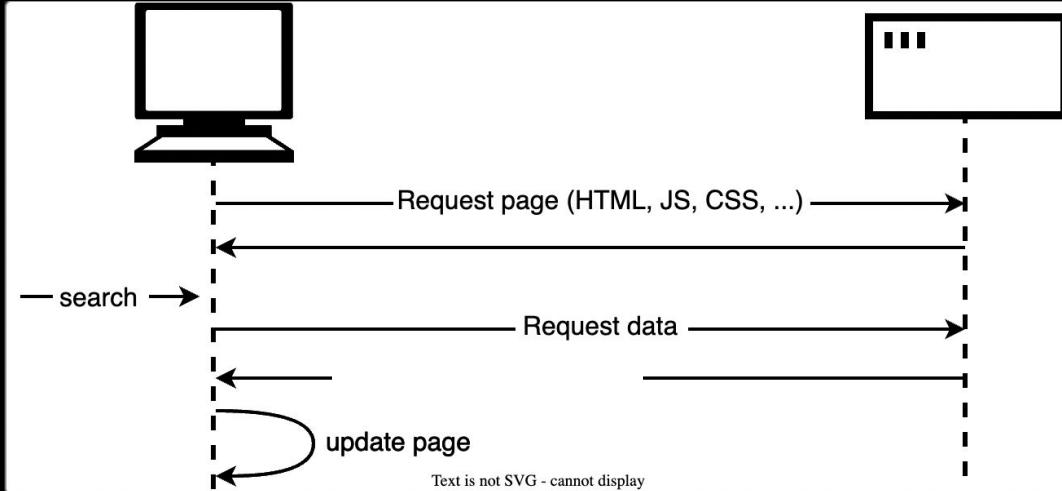
23. REST API / JSON Requests

1. What are *Async Requests*
2. What are *REST APIs*
3. Decoupling Frontend & Backend
4. Routes & HTTP Methods
5. REST Core Concepts
6. First API Todo App
7. API for Fetch Items
8. API for Deleting Items
9. Adding *Complete Item* functionality





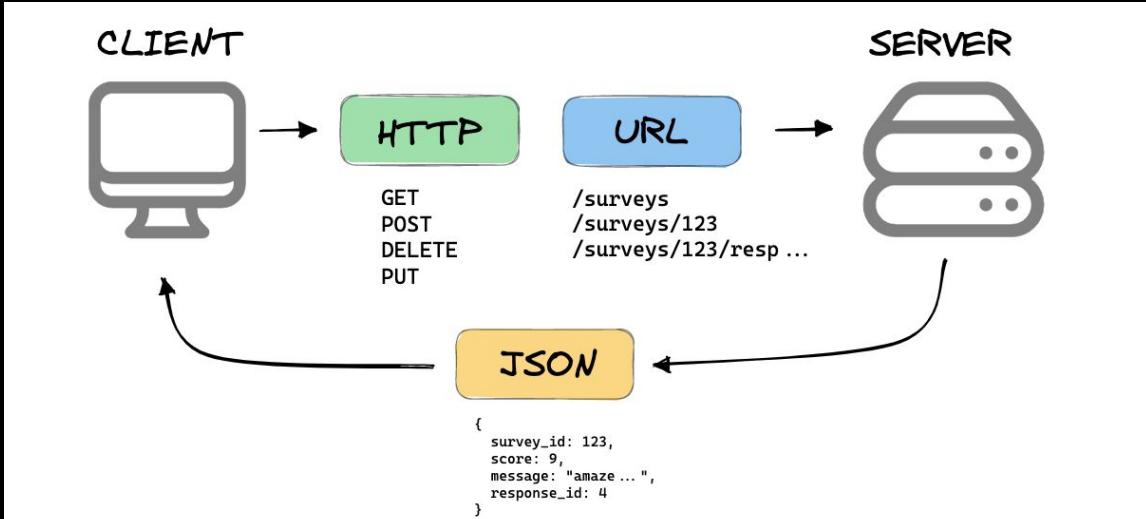
23.1 What are Async Requests



- Async network requests enable web pages to communicate with servers without reloading.
- The client sends JSON requests to the server asynchronously in single-page apps.
- The server processes the request and returns a JSON response.
- The page updates itself dynamically using the received data.



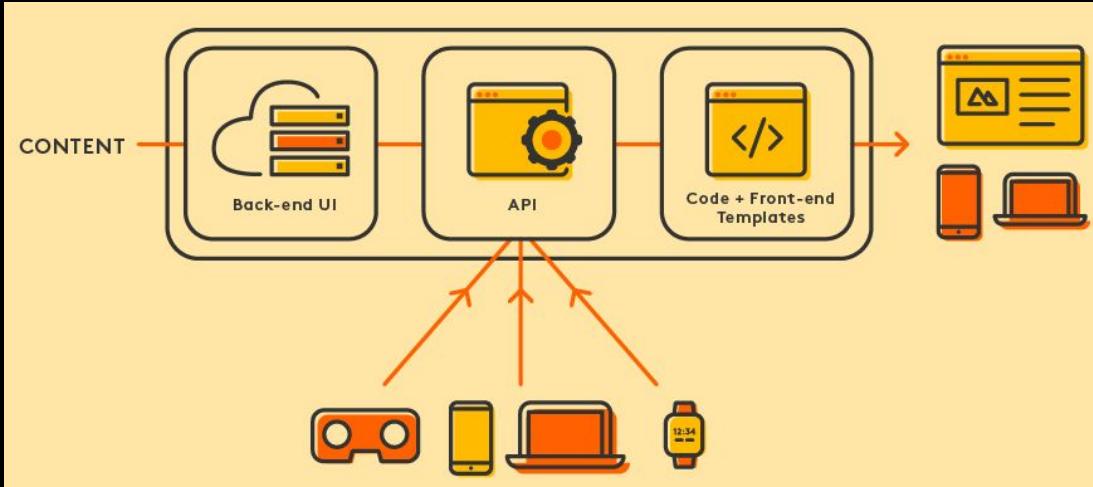
23.2 What are REST APIs



- REST APIs enable communication between clients and servers using HTTP.
- They are mainly identified by a URI.
- They use standard HTTP methods like GET, POST, PUT, and DELETE.
- Data is exchanged in formats like JSON or XML.
- REST APIs are stateless.
- REST APIs allow clients to access and manipulate web resources.



23.3 Decoupling Frontend & Backend



- Separating front-end and back-end allows independent development and scaling.
- REST APIs serve as a communication layer between them.
- Front-end interacts with back-end through standardized RESTful calls.
- Decoupling enhances flexibility and simplifies maintenance.
- REST APIs enable front-end updates without altering back-end code.



23.4 Routes & HTTP Methods

- REST API routes define the endpoints (URLs) where resources can be accessed by clients.
- GET: Retrieves data from the server at the specified route.
- POST: Sends new data to the server to create a resource.
- PUT: Updates or replaces an existing resource at a given route.
- DELETE: Removes a resource from the server at the specified route.
- PATCH: Partially updates an existing resource with new data.

| superheroes | | |
|-------------|--------------------------|--|
| GET | /api/dc/superheroes | Retrieves the collection of superheroes resources. |
| POST | /api/dc/superheroes | Creates a superheroes resource. |
| GET | /api/dc/superheroes/{id} | Retrieves a superheroes resource. |
| PUT | /api/dc/superheroes/{id} | Replaces the superheroes resource. |
| DELETE | /api/dc/superheroes/{id} | Removes the superheroes resource. |
| PATCH | /api/dc/superheroes/{id} | Updates the superheroes resource. |



23.5 REST Core Concepts



- Statelessness: Each request contains all necessary information; the server maintains no client session.
- Uniform Interface: Standardized communication using HTTP methods like GET, POST, PUT, DELETE.
- Client-Server Separation: Independent development of front-end and back-end components.
- Cacheability: Responses indicate if they can be cached to improve performance.
- Layered System: Architecture allows for multiple layers between client and server.
- Code on Demand (Optional): Servers can extend client functionality by sending executable code.



23.6 First API Todo App

1. Copy the following from the previous app:
 - a. Env files
 - b. Package dependencies
 - c. .gitignore
 - d. app.js
2. Changes in env files
 - a. Remove the email data.
 - b. Change DB name to todo-app
3. Create empty controllers and routes folders.
4. Remove excess code from app.js.
5. Have the error controller give out a 404 error message and status.
6. Create a Mongoose model for Todoltem.
7. Create itemsRouter and itemsController for POST /todos request from frontend.
8. Install and setup CORS package in backend.
9. Add express.json() to app.



23.6 First API Todo App

1.

```
.env.development  
.env.example  
.env.production  
app.js  
package.json
```

2.

```
backend > .env.development  
1 MONGO_DB_USERNAME=root  
2 MONGO_DB_PASSWORD=root  
3 MONGO_DB_DATABASE=todo-app-test|
```

```
backend > .env.production
```

```
1 MONGO_DB_USERNAME=root  
2 MONGO_DB_PASSWORD=root  
3 MONGO_DB_DATABASE=todo-app|
```

3.

```
> controllers  
> models  
> node_modules  
> routers
```



23.6 First API Todo App

4.

```
const ENV = process.env.NODE_ENV || 'production'
require('dotenv').config({
  path: `.${ENV}`
});

// External Module
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");

// Local Module
const errorController = require("./controllers/errorController");
const itemsRouter = require("./routers/itemsRouter");

const MONGO_DB_URL =
  `mongodb+srv://${process.env.MONGO_DB_USERNAME}:${process.env.MONGO_DB_PASSWORD}@kgcluster.ie6mb.mongodb.net/${process.env.MONGO_DB_DATABASE}`;

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(itemsRouter);
app.use(errorController.get404);

const PORT = process.env.PORT || 3000;
mongoose.connect(MONGO_DB_URL).then(() => {
  app.listen(PORT, () => {
    console.log(`Server running at: http://localhost:\${PORT}`);
  });
});
```



23.6 First API Todo App

backend > models > `TodoItem.js` > ...

6.

```
1 const mongoose = require("mongoose");
2
3 const todoItemSchema = new mongoose.Schema({
4   task: {
5     type: String,
6     required: true
7   },
8   date: {
9     type: Date,
10    required: true
11 },
12   completed: {
13     type: Boolean,
14     default: false
15 },
16   createdAt: {
17     type: Date,
18     default: Date.now
19 }
20});
```

5.

backend > controllers > `errorController.js` > ...

```
1 exports.get404 = (req, res, next) => {
2   res.status(404).json({ message: 'Page Not Found' });
3 };
```



23.6 First API Todo App

7.

```
const express = require("express");
const itemsController = require("../controllers/itemsController");
const itemsRouter = express.Router();

itemsRouter.post("/todos", itemsController.postItem);

module.exports = itemsRouter;

exports.postItem = async (req, res, next) => {
  try {
    console.log(req.body);
    const todoItem = new TodoItem(req.body);
    const item = await todoItem.save();
    res.json(item);
  } catch (err) {
    next(err);
  }
};
```



23.6 First API Todo App

8,9.

```
const bodyParser = require("body-parser");
const cors = require("cors");

app.use(bodyParser.urlencoded({ extended: true }));
app.use(cors());
app.use(express.json());
app.use(itemsRouter);
app.use(errorController.get404);
```



23.7 API for Fetch Items

```
itemsRouter.get("/todos", itemsController.getItems);
itemsRouter.post("/todos", itemsController.postItem);
```

```
exports.getItems = async (req, res, next) => {
  try {
    const items = await TodoItem.find();
    res.json(items);
  } catch (err) {
    next(err);
  }
};
```

```
export const todoItemToClientModel = (serverItem) => {
  return {
    id: serverItem._id,
    todoText: serverItem.task,
    todoDate: serverItem.date
  }
}
```

```
const formattedDate = new Date(todoDate).toLocaleDateString('en-US', {
  year: 'numeric',
  month: 'long',
  day: 'numeric'
});
```



23.8 API for Deleting Items

```
itemsRouter.get("/todos", itemsController.getItems);
itemsRouter.post("/todos", itemsController.postItem);
itemsRouter.delete("/todos/:id", itemsController.deleteItem);
```

```
exports.deleteItem = async (req, res, next) => {
  try {
    const { id } = req.params;
    await TodoItem.findByIdAndDelete(id);
    res.json({ id, message: "Item deleted" });
  } catch (err) {
    next(err);
  }
};
```



23.9 Improving UI Elements

1. Remove bootstrap
2. Add tailwind to the project.
3. Create the whole app using tailwind classes.



23.10 Adding Complete Item functionality

1. Add a UI element to mark the item complete.
2. Add a component state based on which items will be striked through, default value should come from server.
3. Add a toggleComplete handler in Todoltem that sends a PATCH request to update the completed flag and expects the updated item in return. Also the value of state must be updated based on the final value.
4. Add a route and API in the backend to update the todoltem.



23.10 Adding Complete Item functionality

1,2.

```
<input
  type="checkbox"
  checked={isComplete}
  onChange={toggleComplete}
  className="w-5 h-5 rounded border-gray-300 text-blue-600 focus:ring-blue-500"
/>
```

```
const TodoItem = ({ id, todoText, todoDate, completed }) => {
  const { deleteTodoItem } = useContext(TodoItemsContext);
  const [isComplete, setIsComplete] = useState(completed);
```

```
<div className={`flex flex-col ${isComplete ? 'text-gray-400 line-through' : 'text-gray-700'}`}>
  <span className="font-medium">{todoText}</span>
  <span className="text-sm text-gray-500">{formattedDate}</span>
</div>
```



23.10 Adding Complete Item functionality

3. const toggleComplete = () => {
 fetch(`http://localhost:3000/todos/\${id}` , {
 method: 'PATCH',
 headers: { 'Content-Type': 'application/json' },
 body: JSON.stringify({ completed: !isComplete }) })
 .then(res => res.json())
 .then(data => {
 setIsComplete(data.completed);
 })
 .catch(err => console.log(err));
}



23.10 Adding Complete Item functionality

4.

```
itemsRouter.get("/todos", itemsController.getItems);
itemsRouter.post("/todos", itemsController.postItem);
itemsRouter.delete("/todos/:id", itemsController.deleteItem);
itemsRouter.patch("/todos/:id", itemsController.updateItem);

exports.updateItem = async (req, res, next) => {
  try {
    const { id } = req.params;
    const { completed } = req.body;
    const updatedItem = await TodoItem.findByIdAndUpdate(
      id,
      { completed: completed },
      { new: true }
    );

    res.json(updatedItem);
  } catch (err) {
    next(err);
  }
};
```



23.11 Deploying React App

```
prashantjain@Prashants-MacBook-Pro frontend % az extension add --name staticwebapp
No stable version of 'staticwebapp' to install. Preview versions allowed.
The installed extension 'staticwebapp' is in preview.
```

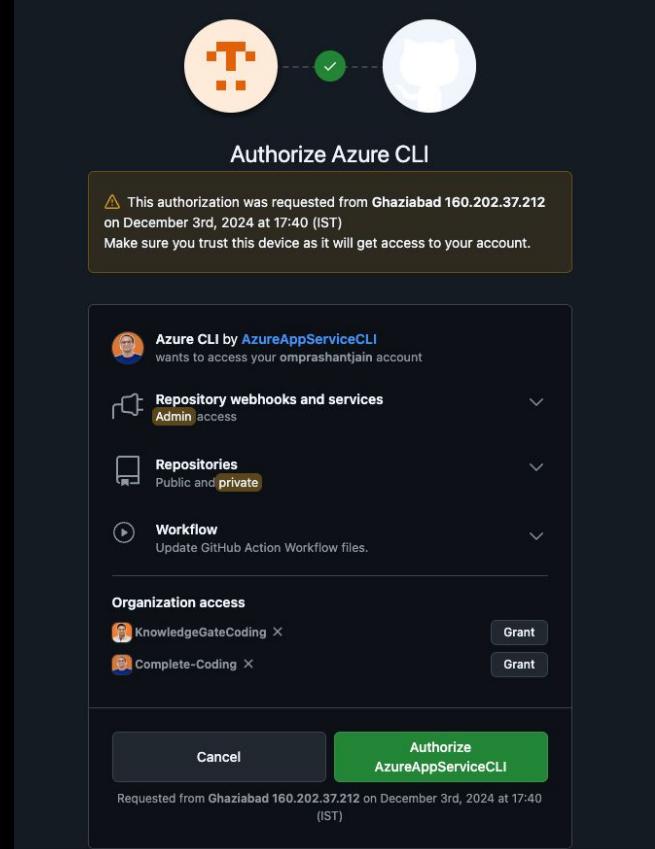
```
prashantjain@Prashants-MacBook-Pro frontend % npm run build
> 6-todo-ui-bootstrap@0.0.0 build
> vite build

vite v5.4.3 building for production...
✓ 38 modules transformed.
dist/index.html      0.46 kB | gzip:  0.30 kB
dist/assets/index-kY9ww-QL.css  9.31 kB | gzip:  2.58 kB
dist/assets/index-DGJMYW8J.js  147.18 kB | gzip: 47.49 kB
✓ built in 384ms
```



23.11 Deploying React App

```
prashantjain@Prashants-MacBook-Pro MERNLiveFrontend % az staticwebapp create \
--name MERNLiveFrontend \
--resource-group omprashantjain_rg_7471 \
--location "Central US" \
--source https://github.com/Complete-Coding/MERNLiveFrontend \
--branch main \
--app-location "/" \
--output-location "dist" \
--login-with-github
Please navigate to https://github.com/login/device and enter the user code 8D2F-CE49 to activate and retrieve your github personal access token
Waiting up to '14' minutes for activation
{
  "allowConfigFileUpdates": true,
  "branch": "main",
  "buildProperties": null,
  "contentDistributionEndpoint": "https://content-dm1.infrastructure.4.azurestaticapps.net",
  "customDomains": [],
  "databaseConnections": [],
  "defaultHostName": "white-grass-0bb31e810.4.azurestaticapps.net",
  "enterpriseGradeCdnStatus": "Disabled",
  "id": "/subscriptions/e66737a1-e433-403b-aacb-e39a9d24b588/resourceGroups/omprashantjain_rg_7471/providers/Microsoft.Web/staticSites/MERNLiveFrontend",
  "identity": null,
  "keyVaultReferenceIdentity": "SystemAssigned",
  "kind": null,
  "linkedBackends": [],
  "location": "Central US",
  "name": "MERNLiveFrontend",
  "privateEndpointConnections": [],
  "provider": "GitHub",
  "publicNetworkAccess": null,
  "repositoryToken": null,
  "repositoryUrl": "https://github.com/Complete-Coding/MERNLiveFrontend",
  "resourceGroup": "omprashantjain_rg_7471",
  "sku": {
    "capabilities": null,
    "capacity": null,
    "family": null,
    "locations": null,
    "name": "Free",
    "size": null,
    "skuCapacity": null,
    "tier": "Free"
  },
  "stagingEnvironmentPolicy": "Enabled",
  "tags": null,
  "templateProperties": null,
  "type": "Microsoft.Web/staticSites",
  "userProvidedFunctionApps": null
}
```





23.11 Deploying React App

The screenshot shows the VS Code interface with a search bar at the top containing the URL `http://mern-live-backend.azurewebsites.net`. Below the search bar, a dropdown menu lists the result `//mern-live-backend.azurewebsites.net`, which is currently selected. The main pane displays search results for files to include and files to exclude, with a total of 4 results found across 3 files. The results are listed in the following table:

| File | Path | Count |
|---------------|----------------|-------|
| AddTodo.jsx | src/components | 1 |
| LoadItems.jsx | src/components | 1 |
| TodolItem.jsx | src/components | 2 |

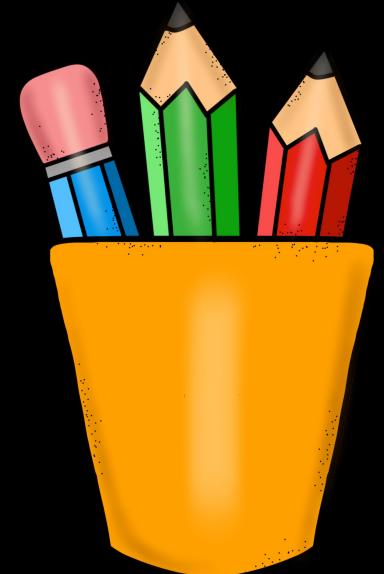
The results show code snippets using the `fetch` API to interact with the deployed backend. The first two results use the full URL `http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos`, while the last two results use a template literal with the URL ``http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos/${i...}``.

```
http://mern-live-backend.azurewebsites.net
//mern-live-backend.azurewebsites.net
files to include
files to exclude
4 results in 3 files - Open in editor
AddTodo.jsx src/components M 1
fetch("http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos", {
LoadItems.jsx src/components M 1
fetch("http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos")
TodolItem.jsx src/components M 2
fetch(`http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos/${i...
fetch(`http://mern-live-backend.azurewebsites.net//mern-live-backend.azurewebsites.net/todos/${i...
```



Revision

1. What are Async Requests
2. What are REST APIs
3. Decoupling Frontend & Backend
4. Routes & HTTP Methods
5. REST Core Concepts
6. First API Todo App
7. API for Fetch Items
8. API for Deleting Items
9. Improving UI Elements
10. Adding Complete Item functionality
11. Deploying React App



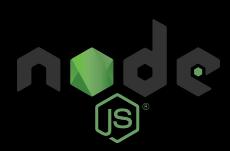


Practise Milestone

Take your todo-app forward:

1. Add other features like editing an item text.
2. Changing the date of completion.







7. NPM & Tools



1. Install Material Icons
2. npm init
3. npm Scripts
4. npm Packages
5. Installing Packages
6. Installing nodemon
7. Using nodemon





React

7.1 Install Material Icons



Material Icon Theme v5.11.1

Philipp Kief

25,491,060

★★★★★ (345)

♥ Sponsor

Material Design Icons for Visual Studio Code

[Set File Icon Theme](#)

[Disable](#) ▾

[Uninstall](#) ▾

Auto Update



React



7.2 npm init

npm init

```
prashantjain@Prashants-Mac-mini user % npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.
```

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.

```
package name: (user) User Backend
Sorry, name can only contain URL-friendly characters and name can no longer contain capital letters.
```

```
package name: (user) user-backend
```

```
version: (1.0.0)
```

```
description: This project will have the backend code of our User project.
```

```
entry point: (user.js) app.js
```

```
test command:
```

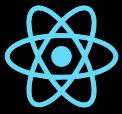
```
git repository:
```

```
keywords:
```

```
author: Complete Coding
```

```
license: (ISC)
```

About to write to /Users/prashantjain/workspace/NodeJS_Complete_YouTube/Chapter 6 – Event Loop/user/package.json:



React

7.3 npm Scripts



```
{  
  "name": "user-backend",  
  "version": "1.0.0",  
  "description": "This project will have the backend code of our User  
project.",  
  "main": "app.js",  
  ▷ Debug  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1",  
    "start": "node app.js",  
    "khul-ja-sim-sim": "node app.js"  
  },  
  "author": "Complete Coding",  
  "license": "ISC"  
}
```

npm start

npm run khul-ja-sim-sim



7.4 npm Packages



1. npm: Node.js package manager for code sharing.
2. Package: Reusable code or library.
3. package.json: Defines package metadata and dependencies.
4. Versioning: Manages different package versions.
5. Local/Global: Install packages locally or globally.
6. Registry: Public storage for open-source packages.
7. Examples: Express, React, Lodash.





7.5 Installing Packages

`npm install <package-name>`

1. **-save:** Adds the package to the project's dependencies in `package.json`.
2. **-save-dev:** Adds the package to the project's `devDependencies` (used only in development) in `package.json`.
3. **-g:** Installs the package globally, making it available system-wide, not just in a specific project.
4. **-save-exact:** Installs the exact version specified without updating for newer versions.
5. **-force:** Forces npm to fetch and install packages even if they are already installed.



React

7.6 Installing nodemon



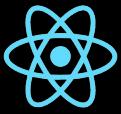
npm install nodemon --save-dev

```
user
  └── node_modules
  ├── app.js
  ├── package-lock.json
  ├── package.json
  └── user.js
      └── user.txt
```

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node app.js",
  "khul-ja-sim-sim": "node app.js"
},
"author": "Complete Coding",
"license": "ISC",
"devDependencies": {
  "nodemon": "^3.1.7"
}
```

npm install

Recreates node_modules



React

7.7 Using nodemon



```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "start": "nodemon app.js",  
  "khul-ja-sim-sim": "node app.js"  
},
```

```
prashantjain@Prashants-Mac-mini user % nodemon app.js
```

```
zsh: command not found: nodemon
```

```
prashantjain@Prashants-Mac-mini user % npm start
```

```
> user-backend@1.0.0 start
```

```
> nodemon app.js
```

```
[nodemon] 3.1.7
```

```
[nodemon] to restart at any time, enter `rs`
```

```
[nodemon] watching path(s): .*
```

```
[nodemon] watching extensions: js,mjs,cjs,json
```

```
[nodemon] starting `node app.js`
```

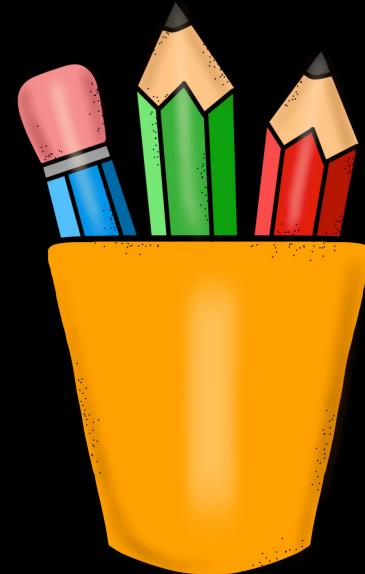
```
Server running on address http://localhost:3001
```

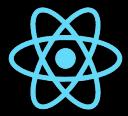
```
npm install nodemon -g
```



Revision

1. Install Material Icons
2. npm init
3. npm Scripts
4. npm Packages
5. Installing Packages
6. Installing nodemon
7. Using nodemon





React



8. Errors & Debugging

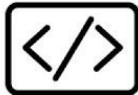
1. Types of Errors
2. Syntax Errors
3. Runtime Errors
4. Logical Errors
5. Using the Debugger
6. Debugger with Async Code
7. Restart Debug with nodemon





8.1 Types of Errors

1



Syntax
Error

2



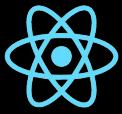
Logical
Error

3



Run-Time
Error

1. **Syntax Error:** An **error** in the **code's structure**, causing it to not compile or run (e.g., missing semicolon).
2. **Logical Error:** The code runs but **produces incorrect results** due to **faulty logic** (e.g., wrong formula).
3. **Runtime Error:** An **error** that occurs while the program is running, often due to **invalid operations**



React

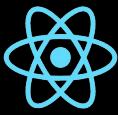
8.2 Syntax Errors

```
// Missing parenthesis in function call  
console.log("Hello, world"
```

```
// Unclosed string literal  
let message = "Welcome to Node.js; ~
```

```
// Improper use of reserved keywords  
let new = 5;
```

```
// Incorrect variable declaration (const needs an initial value)  
const myVar;
```



React

8.3 Runtime Errors

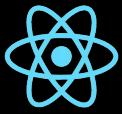
```
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
  at ServerResponse.setHeader (node:_http_outgoing:699:11)
  at IncomingMessage.<anonymous> (/Users/prashantjain/workspace/Test Project/node/app.js:44:11)
  at IncomingMessage.emit (node:events:532:35)
  at endReadableNT (node:internal/streams/readable:1696:12)
  at process.processTicksAndRejections (node:internal/process/task_queues:82:21) {
  code: 'ERR_HTTP_HEADERS_SENT'
}
```

```
// Reference Error (x is not defined)
console.log(x);

// Type Error (num is not a function)
let num = 10;
num();

// Invalid JSON parse (SyntaxError)
let jsonString = "{ name: 'John' }"; // Invalid JSON (single quotes)
JSON.parse(jsonString);

// File not found error (fs module)
const fs = require('fs');
fs.readFileSync('nonexistentFile.txt'); // Throws Error: ENOENT (file not found)
```



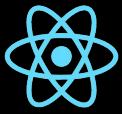
React

8.4 Logical Errors

```
let x = 5;  
if (x = 10) { // Assignment instead of comparison  
|   console.log("x is 10"); // Incorrectly prints this  
}  
  
// Output: x is 10
```

```
let arr = [1, 2, 3, 4, 5];  
for (let i = 0; i <= arr.length; i++) {  
|   console.log(arr[i]); // Prints undefined at the end of the loop  
}  
  
// Output: 1, 2, 3, 4, 5, undefined
```

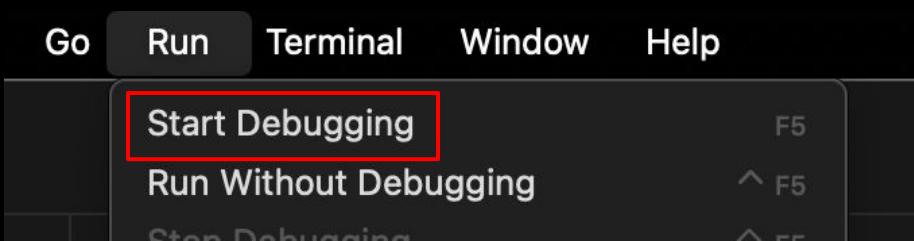
```
let num = "10";  
console.log(num + 5); // Expected result: 15, prints 105  
  
// Output: 105
```



React

8.5 Using the Debugger

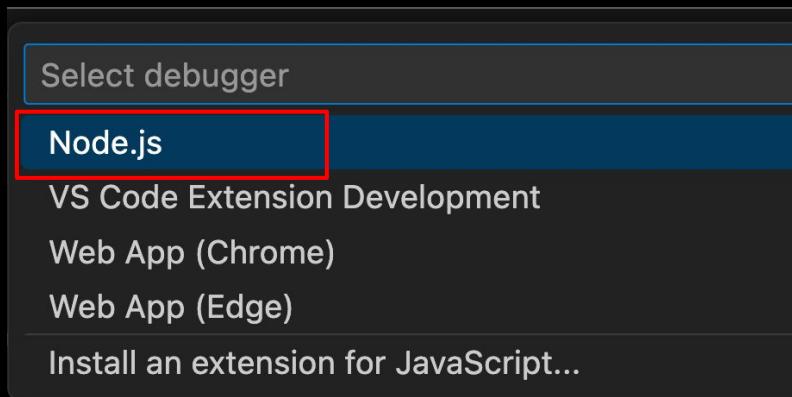
Step 1



Step 3: Put a breakpoint

```
1 let x = 5;
2 if (x = 10) { // Assignment instead of comparison
3   console.log("x is 10"); // Incorrectly prints this
4 }
5
```

Step 2





8.5 Using the Debugger

Step 4: Use the tools



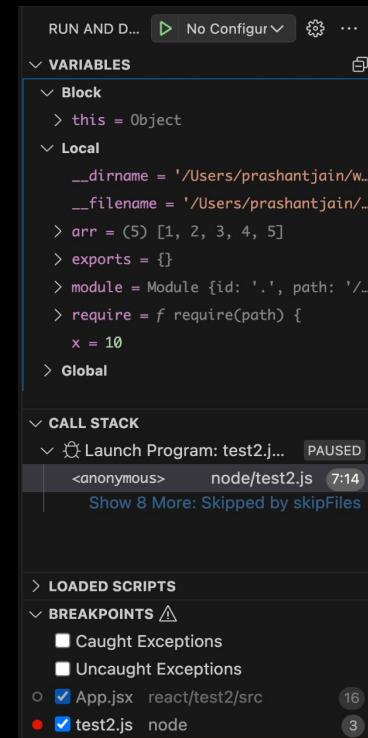
Step 5: Hover

test2.js

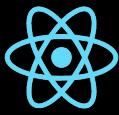
```
let arr = [1, 2, 3, 4, 5];
for (let i = 0; i <= arr.length; i++) {
    console.log(arr[i]); // Prints undefined
}
```

(5) [1, 2, 3, 4, 5]
0 = 1
1 = 2
2 = 3
3 = 4
4 = 5
length = 5
> [[Prototype]] = Array(0)
> [[Prototype]] = Object

Hold Option key to switch to editor language hover



Step 6: Debug Panel



React

8.5 Using the Debugger

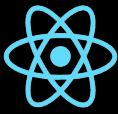
Step 7: Using Debug Console

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/opt/homebrew/bin/node ./node/test2.js
```

```
→ x  
10  
→ x++  
10  
→ x === 15  
false
```

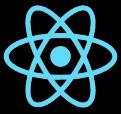
>



React

8.6 Debugger with Async Code

- 31 console.log("Here");
- 32 req.on("end", () => {
- 33 const parsedBody = Buffer.concat(body).toString();
- 34 console.log(parsedBody);
- 35 const params = new URLSearchParams(parsedBody);
- 36 const jsonObject = {};
- 37 for (const [key, value] of params.entries()) {
- 38 jsonObject[key] = value;
- 39 }
- 40 const jsonString = JSON.stringify(jsonObject);
- 41 console.log(jsonString);
- 42 // Async Operation
- 43 fs.writeFile("user-details.txt", jsonString, error => {
- 44 res.setHeader("Location", "/");
- 45 res.statusCode = 302;
- 46 return res.end();
- 47 });
- 48 });
- 49 }



React

Run Terminal Window

Start Debugging
Run Without Debugging
Stop Debugging
Restart Debugging

Open Configurations
Add Configuration...

Step Over
Step Into

8.7 Restart Debug with nodemon



```
"version": "0.2.0",
"configurations": [
  {
    "type": "node",
    "request": "launch",
    "name": "Launch Program",
    "skipFiles": [
      "<node_internals>/**"
    ],
    "program": "${workspaceFolder}/node/test2.js",
    "restart": true,
    "runtimeExecutable": "nodemon",
    "console": "integratedTerminal"
  }
]
```



Practise Set

Debug and fix Syntax, Runtime and Logical Errors

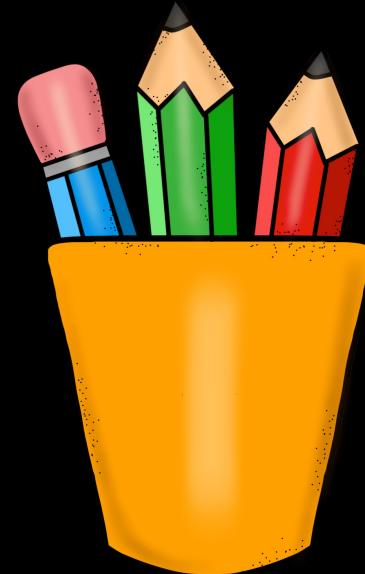
```
function calculateArea(width, height {  
    return width + height;  
}  
  
let width = 10 height = 5;  
  
if (area > 100) {  
    console.log("The area is large.");  
} else {  
    console.log("The area is small.");  
}  
  
if (width + height > 100) {  
    console.log("Area is greater than or equal to 100");  
}
```

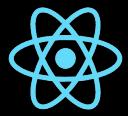




Revision

1. Types of Errors
2. Syntax Errors
3. Runtime Errors
4. Logical Errors
5. Using the Debugger
6. Debugger with Async Code
7. Restart Debug with nodemon





React



React

12. Dynamic UI using EJS

1. Need of Dynamic UI
2. Sharing using Global Variables
3. What is EJS
4. Installing EJS
5. Using EJS Templates
6. Working with Partials



12.1 Need of Dynamic UI



- **Personalized Content:** Tailors responses based on user profiles, preferences, or behaviors to enhance user experience.
- **Dynamic Data Delivery:** Provides real-time information that updates with each request, such as live scores or stock prices.
- **Security and Access Control:** Delivers different content based on user authentication and authorization levels.
- **Localization and Internationalization:** Adjusts responses to accommodate different languages, cultures, or regional settings.
- **API Versatility:** Supports multiple client types (web, mobile, IoT) by providing appropriate data formats and structures.



12.2 Sharing using Global Variables <%

```
const homes = [];

// POST path
router.post('/add-home', (req, res, next) => {
  homes.push({houseName: req.body.houseName});
  res.redirect('/');
});

exports.hostRouter = router;
exports.homes = homes;
```

Variables are shared across
users

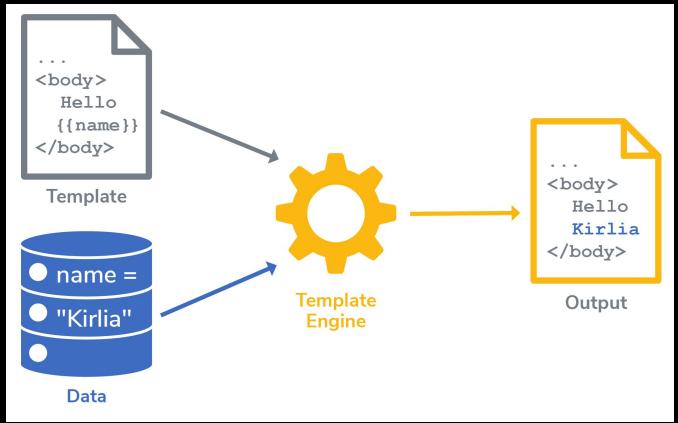
```
const {homes} = require('../routes/host');

userRouter.get('/', (req, res, next) => {
  console.log('homes', homes);
  res.sendFile(path.join(rootDir, 'views', 'user.html'));
});
```

12.3 What is EJS

<%

- **HTML with JS:** EJS lets you embed **JavaScript** code within **HTML**.
- **Simple Syntax:** Uses `<% %>` for control flow and `<%= %>` for output.
- **Easy to Learn:** Familiar to those who know **HTML** and **JavaScript**.
- **Template Reuse:** Supports **partials** for reusing code snippets.
- **Flexible Logic:** Allows full **JavaScript** expressions in templates.



12.4 Installing EJS



```
prashantjain@Prashants-Mac-mini air-bnb % npm install --save ejs
```

```
added 6 packages, and audited 232 packages in 879ms
```

```
const app = express();
```

```
app.set('view engine', 'ejs');  
app.set('views', 'views');
```

12.4 Installing EJS



app.set(name, value)

Assigns setting `name` to `value`. You may store any value that you want, but certain names can be used to configure the behavior of the server. These special names are listed in the [app settings table](#).

Calling `app.set('foo', true)` for a Boolean property is the same as calling `app.enable('foo')`. Similarly, calling `app.set('foo', false)` for a Boolean property is the same as calling `app.disable('foo')`.

Retrieve the value of a setting with `app.get()`.

```
app.set('title', 'My Site')
app.get('title') // "My Site"
```

| | | | |
|--------------------------|-----------------|---|---|
| <code>views</code> | String or Array | A directory or an array of directories for the application's views. If an array, the views are looked up in the order they occur in the array. | <code>process.cwd() + '/views'</code> |
| <code>view cache</code> | Boolean | Enables view template compilation caching. NOTE: Sub-apps will not inherit the value of this setting in production (when 'NODE_ENV' is "production"). | <code>true</code> in production, otherwise <code>undefined</code> . |
| <code>view engine</code> | String | The default engine extension to use when omitted. NOTE: Sub-apps will inherit the value of this setting. | N/A (<code>undefined</code>) |

12.5 Using EJS Templates



✓ views

404.html

add-home.html

input.css

<% user.ejs

```
// Local Modules
const {homes} = require('../routes/host');

userRouter.get('/', (req, res, next) => {
  res.render('user', { homes: homes });
});
```

```
<main class="container mx-auto bg-white shadow-lg rounded-lg p-8 mt-10 max-w-xl">
  <h2 class="text-5xl text-gray-800 font-bold text-center mb-6">
    Available Homes
  </h2>
  <ul class="list-disc list-inside space-y-2">
    <% homes.forEach(function(home) { %>
      <li class="text-gray-700 hover:text-red-500 transition-colors duration-300 text-2xl">
        <%= home.houseName %>
      </li>
    <% }); %>
  </ul>
</main>
```

12.6 Working with Partials



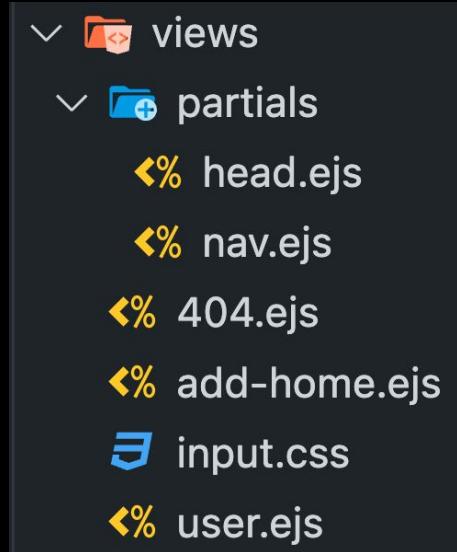
```
views
  partials
    head.ejs
    nav.ejs
    404.ejs
    add-home.ejs
    input.css
    user.ejs
```

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title><%= title %></title>
7      <link rel="stylesheet" href="/output.css" />
8
```

12.6 Working with Partials



```
1 <header class="■bg-red-500 ■text-white p-4 shadow-md">
2   <nav class="container mx-auto flex justify-between items-center">
3     <ul class="flex space-x-4">
4       <li>
5         <a
6           href="/"
7           class="■hover:bg-red-400 py-2 px-4 rounded transition duration-300"
8           >Go to Home</a>
9       >
10      </li>
11      <li>
12        <a
13          href="/host/add-home"
14          class="■hover:bg-red-400 py-2 px-4 rounded transition duration-300"
15          >Add Home</a>
16        >
17      </li>
18    </ul>
19  </nav>
20 </header>
```



12.6 Working with Partials



airbnb > views > 404.ejs > ?

```
<%- include('partials/head') %>
</head>
<body class="■bg-gray-100 font-sans">
  <%- include('partials/nav') %>
  <main class="container mx-auto mt-20 text-center">
    <h2 class="text-6xl font-bold ■text-red-500 mb-4">404</h2>
    <p class="text-2xl □text-gray-700 mb-8">Oops! Page Not Found</p>
    <a href="/" class="■bg-red-500 ■text-white py-2 px-6 rounded-lg
      ■hover:bg-red-600 transition duration-300">Go Back Home</a>
  </main>
</body>
</html>
```

```
app.use((req, res, next) => {
  res.status(404).render('404', {title: 'Page Not Found'});
});
```

12.6 Working with Partials

<%

```
userRouter.get('/', (req, res, next) => {
  res.render('user', { homes: homes, title: 'Available Homes' });
});

<%-- include('partials/head') %>
</head>
<body class="■bg-gray-100 font-sans">
  <%-- include('partials/nav') %>
  <main class="container mx-auto ■bg-white shadow-lg rounded-lg p-8 mt-10 max-w-xl">
    <h2 class="text-5xl □text-gray-800 font-bold text-center mb-6">
      Available Homes
    </h2>
```



React

Practise Set

Reuse the app from the last assignment

1. Add more fields in the add home page like price per night, location, rating, photo.
2. Design the home card to show all of this information
3. Make the selected tab active on top.

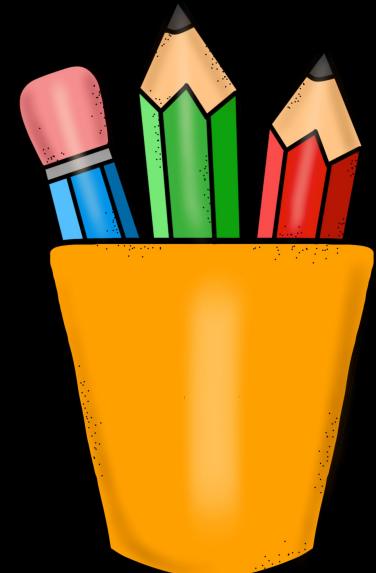




React

Revision

1. Need of Dynamic UI
2. Sharing using Global Variables
3. What is EJS
4. Installing EJS
5. Using EJS Templates
6. Working with Partials





13. Model View Controller

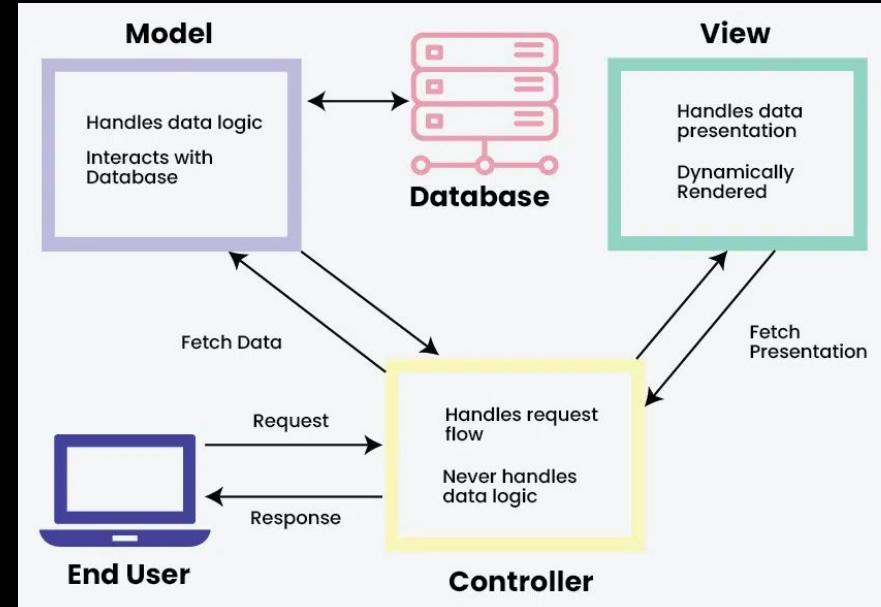
1. Separation of Concerns
2. Adding First Controller
3. Adding All Controllers
4. Adding 404 Controller
5. Adding Models
6. Writing data to Files
7. Nodemon causing problems
8. Reading data from Files





13.1 Separation of Concerns

- **MVC stands for Model-View-Controller:** A software architectural pattern for developing user interfaces.
- **Model:** Manages the data and business logic of the application.
- **View:** Handles the display and presentation of data to the user.
- **Controller:** Processes user input, interacts with the Model, and updates the View accordingly.
- Routes are a part of Controllers.
- **Purpose:** MVC separates concerns within an application, making it easier to manage and scale.





13.2 Adding First Controller

File tree:

- controllers
 - homes.js
- node_modules
- public
- routes
 - hostRouter.js
 - userRouter.js
- utils
- views
 - partials
 - 404.ejs
 - addHome.ejs
 - home.ejs
 - homeAdded.ejs
 - input.css
 - app.js
 - package-lock.json
 - package.json
 - tailwind.config.js

```
js homes.js x
node > Chapter 12 - Dynamic > controllers > js homes.js > ...
3 exports.getAddHome = (req, res, next) => {
4   res.render("addHome", {
5     pageTitle: "Add Home to airbnb",
6     currentPage: "addHome",
7   });
8 };
```

```
js hostRouter.js x
node > Chapter 12 - Dynamic > routes > js hostRouter.js > ...
1 // External Module
2 const express = require('express');
3 const homesController = require('../controllers/homes');
4
5 const hostRouter = express.Router();
6
7 hostRouter.get("/add-home", homesController.getAddHome);
```



13.3 Adding All Controllers

js homes.js ×

```
node > Chapter 12 - Dynamic > controllers > js homes.js > ...
1 const registeredHomes = [];
2
3 exports.getAddHome = (req, res, next) => {
4   res.render("addHome", {
5     pageTitle: "Add Home to airbnb",
6     currentPage: "addHome",
7   });
8 };
9
10 exports.postAddHome = (req, res, next) => {
11   console.log("Home Registration successful for:", req.body);
12   registeredHomes.push(req.body);
13   res.render("homeAdded", {
14     pageTitle: "Home Added Successfully",
15     currentPage: "homeAdded",
16   });
17 };
18
19 exports.getHomes = (req, res, next) => {
20   console.log(registeredHomes);
21   res.render("home", {
22     registeredHomes: registeredHomes,
23     pageTitle: "airbnb Home",
24     currentPage: "Home",
25   });
26 };
```



13.3 Adding All Controllers

JS hostRouter.js ×

node > Chapter 12 - Dynamic > routes > JS hostRouter.js > ...

```
1 // External Module
2 const express = require('express');
3 const homesController = require('../controllers/homes');
4
5 const hostRouter = express.Router();
6
7 hostRouter.get("/add-home", homesController.getAddHome);
8
9 hostRouter.post("/add-home", homesController.postAddHome);
10
11 exports.hostRouter = hostRouter;
```



13.3 Adding All Controllers

JS userRouter.js ×

node > Chapter 12 - Dynamic > routes > JS userRouter.js > ...

```
1 // External Module
2 const express = require('express');
3 const userRouter = express.Router();
4 const homesController = require('../controllers/homes');
5
6 userRouter.get("/", homesController.getHomes);
7
8 module.exports = userRouter;
```



13.4 Adding 404 Controller

```
controllers
  error.js
  homes.js
> node_modules
> public
< routes
  hostRouter.js
  userRouter.js
> utils
< views
  partials
    404.ejs
    addHome.ejs
    home.ejs
    homeAdded.ejs
    input.css
  app.js
  package-lock.json
  package.json
  tailwind.config.js
```

```
error.js  x
node > Chapter 12 - Dynamic > controllers > error.js > ...
1  exports.get404 = (req, res, next) => {
2    res.status(404).render('404', {pageTitle: 'Page Not Found', currentPage: '404'});
3  }

//Local Module
const userRouter = require("./routes/userRouter")
const {hostRouter} = require("./routes/hostRouter")
const rootDir = require("./utils/pathUtil");
const errorController = require("./controllers/error");

const app = express();

app.set('view engine', 'ejs');
app.set('views', 'views');

app.use(express.urlencoded());
app.use(userRouter);
app.use("/host", hostRouter);

app.use(express.static(path.join(rootDir, 'public')))

app.use(errorController.get404);
```



13.5 Adding Models

The image shows a code editor interface with a sidebar and a main panel. The sidebar on the left lists the project structure:

- controllers
- models
- node_modules
- public
- routes
- utils
- views
- app.js
- package-lock.json
- package.json
- tailwind.config.js

The main panel shows the content of the file `home.js`:

```
JS home.js x
node > Chapter 12 - Dynamic > models > JS home.js > ...
1 // fake database
2 const registeredHomes = [];
3
4 module.exports = class Home {
5   constructor(houseName, price, location, rating, photoUrl) {
6     this.houseName = houseName;
7     this.price = price;
8     this.location = location;
9     this.rating = rating;
10    this.photoUrl = photoUrl;
11  }
12
13  save() {
14    registeredHomes.push(this);
15  }
16
17  static fetchAll() {
18    return registeredHomes;
19  }
20}
```



13.5 Adding Models

js homes.js x

```
node > Chapter 12 - Dynamic > controllers > js homes.js > ...
1 const Home = require("../models/home");
2
3 exports.getAddHome = (req, res, next) => {
4   res.render("addHome", {
5     pageTitle: "Add Home to airbnb",
6     currentPage: "addHome",
7   });
8 };
9
10 exports.postAddHome = (req, res, next) => {
11   const { houseName, price, location, rating, photoUrl } = req.body;
12   const home = new Home(houseName, price, location, rating, photoUrl);
13   home.save();
14   res.render("homeAdded", {
15     pageTitle: "Home Added Successfully",
16     currentPage: "homeAdded",
17   });
18 };
19
20 exports.getHomes = (req, res, next) => {
21   const homes = Home.fetchAll();
22   res.render("home", {
23     registeredHomes: homes,
24     pageTitle: "airbnb Home",
25     currentPage: "Home",
26   });
27 };
```



13.6 Writing data to Files

```
> controllers
└── data
    └── homes.json
└── models
    └── home.js
> node_modules
> public
> routes
└── utils
    └── pathUtil.js
> views
    └── app.js
└── package-lock.json
└── package.json
└── tailwind.config.js
```

```
1 const fs = require('fs');
2 const path = require('path');
3 const rootDir = require('../utils/pathUtil');
4
5 // fake database
6 const registeredHomes = [];
7
8 module.exports = class Home {
9     constructor(houseName, price, location, rating, photoUrl) {
10         this.houseName = houseName;
11         this.price = price;
12         this.location = location;
13         this.rating = rating;
14         this.photoUrl = photoUrl;
15     }
16
17     save() {
18         registeredHomes.push(this);
19         const filePath = path.join(rootDir, 'data', 'homes.json');
20         fs.writeFile(filePath, JSON.stringify(registeredHomes), (err) => {
21             console.log(err);
22         });
23     }
24
25     static fetchAll() {
26         return registeredHomes;
27     }
28 }
```



13.6 Writing data to Files

homes.json X

node > Chapter 12 - Dynamic > data > homes.json > ...

```
1  [
2  {
3      "houseName": "Utsav",
4      "price": "999",
5      "location": "Ghaziabad",
6      "rating": "5",
7      "photoUrl": "https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.airbnb.com%2Fstays%2Fdesign&psig=A0vVaw259nAPAVlXwRuraoxPS3u7&ust=1729404860631000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTC0iViaXlmYkDFQAAAAAdAAAAABAE"
```

13.7 Nodemon causing problems

```
> controllers
< data
  homes.json
> models
> node_modules
> public
> routes
< utils
  pathUtil.js
> views
  app.js
  nodemon.json
  package-lock.json
  package.json
  tailwind.config.js
```

 nodemon.json ×

node > Chapter 12 - Dynamic >  nodemon.json > ...

```
1  {
2    "watch": ["."],
3    "ext": "js,json,ejs",
4    "ignore": ["node_modules/", "data/"],
5    "exec": "node app.js"
6 }
```



13.8 Reading data from Files

```
static fetchAll() {  
  const filePath = path.join(rootDir, 'data', 'homes.json');  
  const fileContent = fs.readFile(filePath, (err, data) => {  
    if (err) {  
      return [];  
    }  
    return JSON.parse(data);  
  });  
}
```

```
TypeError: /Users/prashantjain/workspace/Test Project/node/Chapter 12 - Dynamic/views/home.ejs:10  
8|           </h2>  
9|           <div class="flex flex-wrap justify-center gap-6">  
>> 10|             <% registeredHomes.forEach(home => { %>  
11|               <div class="bg-white rounded-lg shadow-md overflow-hidden hover:shadow-lg transi  
12|                 
```

```
Cannot read properties of undefined (reading 'forEach')  
at eval (eval at compile (/Users/prashantjain/workspace/Test Project/node/Chapter 12 - Dynamic  
at home (/Users/prashantjain/workspace/Test Project/node/Chapter 12 - Dynamic/node_modules/ejs  
at tryHandleCache (/Users/prashantjain/workspace/Test Project/node/Chapter 12 - Dynamic/node_m  
at exports.renderFile [as engine] (/Users/prashantjain/workspace/Test Project/node/Chapter 12 - Dynamic/node_modules/ejs/lib/ejs.js:69:5)  
at Object.exports.renderFile (/Users/prashantjain/workspace/Test Project/node/Chapter 12 - Dynamic/node_modules/ejs/lib/ejs.js:316:10)
```



13.8 Reading data from Files

```
save() {  
  this.fetchAll((registeredHomes) => {  
    registeredHomes.push(this);  
    const filePath = path.join(rootDir, 'data', 'homes.json');  
    fs.writeFile(filePath, JSON.stringify(registeredHomes), (err) => {  
      console.log(err);  
    });  
  });  
  
  static fetchAll(callback) {  
    const filePath = path.join(rootDir, 'data', 'homes.json');  
    fs.readFile(filePath, (err, data) => {  
      let homes = [];  
      if (!err) {  
        homes = JSON.parse(data);  
      }  
      callback(homes);  
    });  
  }  
}
```

```
exports.getHomes = (req, res, next) => {  
  Home.fetchAll((homes) => {  
    res.render("home", {  
      registeredHomes: homes,  
      pageTitle: "airbnb Home",  
      currentPage: "Home",  
    });  
  });  
};
```



Practise Milestone

Take your airbnb forward:

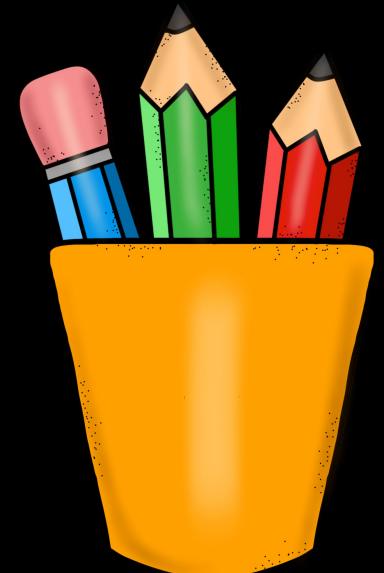
1. Structure the views folder into **host & store** and move the respective view files there.
2. Add more **views** to store like: **home-list**, **home-detail**, **favourite-list**, **reserve**, **bookings**. And to **host** view: **edit-home**, **host-home-list**
3. Improve the **header with navigation** to all pages.
4. Register all the new routes and add dummy views there.
5. Change controllers to **store** and **host** setup.
6. Add Edit and Delete buttons to the **host-home-list** view.
7. Keep the logic for **Edit**, **Delete**, **Favourite pending**.

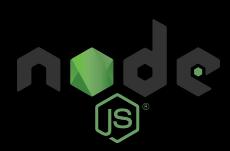




Revision

1. Separation of Concerns
2. Adding First Controller
3. Adding All Controllers
4. Adding 404 Controller
5. Adding Models
6. Writing data to Files
7. Nodemon causing problems
8. Reading data from Files





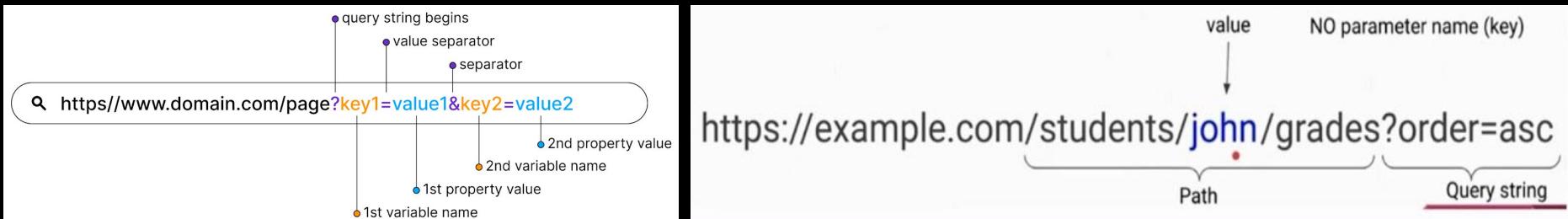
14. Dynamic Path & Model deep-dive

1. What are dynamic paths
2. Adding Home Details Page
3. Showing Real Home Data
4. Adding Favourites Feature
5. Adding Favourites Model
6. Edit-Home: Adding Feature Skeleton
7. Edit-Home: Showing Existing Data
8. Edit-Home: Handing Edit Request
9. Adding Delete Feature
10. Removing Home from Favourites





14.1 What are dynamic paths



- **Path parameters** are variables embedded directly in the URL
path to capture dynamic values, like `/users/:userId` where `:userId` is replaced with a specific user ID.
- **Query parameters** are key-value pairs appended to the URL after a ?, used to send additional data, like `/search?query=nodejs` where `query=nodejs` specifies the search term.



14.2 Adding Home Details Page

1. Add a **details button** in `/homes-list` to go to link path `/homes/:home-id`
2. Add a **random id** to each home in the **data file**.
3. Add a **random id** on home object before **saving** in the **home model**.
4. Add a **route** in the **store router** for `/homes/:home-id`
5. Add a **method in store controller** to get the `home-id` using `req.params` and **log it**, before **sending** out a dummy response with home id.



14.2 Adding Home Details Page

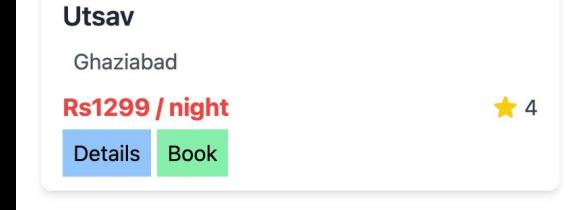
1.

```
views > store > home-list.ejs > ?  
1  <%- include('../partials/head') %>  
2  <body>  
3    <%- include('../partials/nav') %>  
4    <main class="container mx-auto bg-white shadow-lg rounded-lg p-8 mt-10 max-w-6xl">  
5      <div class="flex flex-wrap justify-center gap-6">  
6        <a href="/homes/<%= home.id %>" class="bg-blue-300 p-2">Details</a>  
7        <a href="#" class="bg-green-300 p-2">Book</a>  
8      </div>
```



2.

```
views > store > homes.json > ...  
data > homes.json > ...  
1  [{"id": "1", "houseName": "Utsav", "price": "1299", "location": "Ghaziabad", "rating": "4",  
2    "photoUrl": "asdf"}, {"id": "2", "houseName": "Second House", "price": "999",  
3    "location": "Ghaziabad", "rating": "4", "photoUrl": "asdf"}]
```



3.

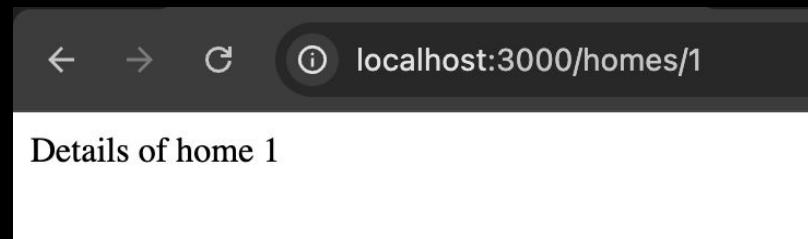
```
models > home.js > ...  
6  module.exports = class Home {  
7  
14  
15    save() {  
16      this.id = Math.random().toString();  
17      Home.fetchAll(registeredHomes) => {
```



14.2 Adding Home Details Page

4.

```
storeRouter.js X storeController.js  
routes > storeRouter.js > ...  
8 storeRouter.get("/", homesController.getIndex);  
9 storeRouter.get("/homes", homesController.getHomes);  
10 // homes/1  
11 storeRouter.get("/homes/:homeId", homesController.getHome);  
12 storeRouter.get("/bookings", homesController.getBookings);
```



5.

```
storeRouter.js X storeController.js  
controllers > storeController.js > ...  
23 exports.getHome = (req, res, next) => {  
24   const homeId = req.params.homeId;  
25   console.log(homeId);  
26   res.send("Details of home " + homeId);  
27 };  
28
```



14.3 Showing Real Home Data

1. Add a static `findById` method in `Home model` that takes a `callback`.
2. Use this `findById` method in the `controller` to load `home details` and `log` them.
3. Now change `the controller`:
 - a. Redirect to `/homes` in case the home is not found, logging an error.
 - b. Rendering the `/home-detail` page with `home data`.
4. Create a `home-detail` page to use the entire page to show all home data.
5. Download from Github:
 - a. The styled `home-detail file`.
 - b. 10 image of `homes` that you can put locally and use.



14.3 Showing Real Home Data

1.

```
storeController.js  home.js  ×  
models > home.js > ...  
 6  module.exports = class Home {  
 32  
 33    static findById(id, callback) {  
 34      this.fetchAll((homes) => {  
 35        const home = homes.find((home) => home.id === id);  
 36        callback(home);  
 37      });  
 38    }  
 39  };
```

```
Server running on address http://localhost:3000  
{  
  id: '1',  
  houseName: 'Utsav',  
  price: '1299',  
  location: 'Ghaziabad',  
  rating: '4',  
  photoUrl: 'asdf'  
}
```

2.

```
exports.getHome = (req, res, next) => {  
  const homeId = req.params.homeId;  
  Home.findById(homeId, (home) => {  
    console.log(home);  
  });  
};
```



14.3 Showing Real Home Data

```
3. exports.getHome = (req, res, next) => {
    const homeId = req.params.homeId;
    Home.findById(homeId, (home) => {
        if (!home) {
            return res.redirect('/homes'); // Redirect if home not found
        }
        res.render("store/home-detail", {
            home: home,
            pageTitle: home.houseName,
            currentPage: "homes",
        });
    });
};
```



14.3 Showing Real Home Data

4.

```
views > store > home-detail.ejs > ↻ ?  
1  <%- include('../partials/head') %>  
2  </head>  
3  <body>  
4    <%- include('../partials/nav') %>  
5    <main class="container mx-auto bg-white shadow-lg rounded-lg p-8 mt-10 max-w-6xl">  
6      <h2 class="text-3xl text-red-500 font-bold text-center mb-6">  
7        Details of <%= home.houseName %>  
8      </h2>  
9  
10     <div class="grid grid-cols-1 md:grid-cols-2 gap-8">  
11       <div class="rounded-lg overflow-hidden">  
12         " class="w-full h-96 object-cover"/>  
13       </div>  
14  
15       <div class="space-y-6">  
16         <div class="border-b pb-4">  
17           <h3 class="text-2xl font-semibold mb-2">Location</h3>  
18           <p class="text-gray-600"><%= home.location %></p>  
19         </div>  
20  
21         <div class="border-b pb-4">  
22           <h3 class="text-2xl font-semibold mb-2">Price</h3>  
23           <p class="text-green-600 text-xl font-bold">$<%= home.price %> / night</p>  
24         </div>  
25  
26         <div class="border-b pb-4">  
27           <h3 class="text-2xl font-semibold mb-2">Rating</h3>  
28           <div class="flex items-center">  
29             <span class="text-yellow-400 text-xl">*</span>  
30             <span class="ml-2 text-lg"><%= home.rating %> / 5</span>  
31           </div>  
32         </div>  
33  
34         <div class="mt-8">  
35           <form action="/favourite" method="post">  
36             <button type="submit" class="bg-red-500 text-white px-8 py-3 rounded-lg hover:bg-  
37               Add to Favorites  
38             </button>  
39           </form>  
40         </div>  
41       </div>  
42     </div>  
43   </main>  
44 </body>  
45 </html>
```

5.

- ⌄ **images**
 - house1.png
 - house2.png
 - house3.png
 - house4.png
 - house5.png
 - house6.png
 - house7.png
 - house8.png



14.4 Adding Favourites Feature

1. Create a partial with a form and a button that submits to /favourites path with a hidden input having home-id value.
2. Add this partial to home-detail page
3. Add this partial to home-list page.
4. Add router for handling POST request to /favourites path.
5. Add a method in store controller to add a home id as favourites, logging the id for now, before redirecting to /favourites path.

node 14.4 Adding Favourites Feature

1.

```
views > partials > favourite.ejs > form
1  <form action="/favourites" method="post">
2    <button type="submit" class="bg-red-500 text-white px-8 py-3 rounded-lg"
3      Add to Favorites
4    </button>
5    <input type="hidden" name="homeId" value="<%= home.id %>">
6  </form>
```

2.

```
views > store > home-detail.ejs > body > main.container.mx-auto.bg-white.shadow-lg.rounded-lg.p-8.mt-10.max-w-6xl > div.grid.grid-cols-1.md:gr
1  <%- include('../partials/head') %>
3  <body>
4    <%- include('../partials/nav') %>
5    <main class="container mx-auto bg-white shadow-lg rounded-lg p-8 mt-10 max-w-6xl">
10       <div class="grid grid-cols-1 md:grid-cols-2 gap-8">
34         <div class="mt-8">
35           <%- include('../partials/favourite') %>
36         </div>
37       </div>
38     </div>
39   </main>
40 </body>
41 </html>
```

node 14.4 Adding Favourites Feature

3.

```
views > store > home-list.ejs > ⌂ ?  
1  <%- include('../partials/head') %>  
3  <body>  
4    <%- include('../partials/nav') %>  
5    <main class="container mx-auto ■bg-white shadow-lg rounded-lg p-8 mt-10 max-w-6xl">  
9      <div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">  
32        <div class="flex gap-3">  
33          <a href="/homes/<%= home.id %>" class="flex-1 ■bg-white ■text-red-500 py  
■hover:text-white border ■border-red-500">View Details</a>  
34          <%- include('../partials/favourite', { home: home }) %>  
35        </div>  
36      </div>  
37    </div>  
38  <% } ) %>  
39  </div>  
40  </main>  
41  </body>  
42 </html>
```

node 14.4 Adding Favourites Feature

4. `storeRouter.post("/favourites", homesController.postFavourites);
storeRouter.get("/favourites", homesController.getFavouriteList);`

5.



A screenshot of a code editor showing a file named `storeController.js`. The code contains a single function `postFavourites` which logs the `homeId` from the request body and then redirects to the '/favourites' route. The code is numbered 54 to 58.

```
controllers > storeController.js > ...
54  exports.postFavourites = (req, res, next) => {
55    const homeId = req.body.homeId;
56    console.log(homeId);
57    res.redirect('/favourites');
58 };
```



14.5 Adding Favourites Model

1. Create a new Model to handle Favourites:
 - a. A static method `getFavourites` that reads the `favourites.json` file and return the ids of all homes marked favourite.
 - b. A static method `addFavourites` that adds the `home-id` to `favourites-id` array if not already there, then updates the file.
2. Use this model in `addFavourites` controller.
3. Use this model in `getFavourites` controller while also fetching details of all homes from Homes Model.
4. Change the UI of favourites page to show new content.



14.5 Adding Favourites Model

1.

```
const fs = require('fs');
const path = require('path');

const rootDir = require('../utils/pathUtil');
const favouritesPath = path.join(rootDir, "data", "favourites.json");

module.exports = class Favourites {
  static addFavourite(homeId) {
    this.getFavourites((favourites) => {
      if (favourites.includes(homeId)) {
        console.log("Home already in favourites");
      } else {
        favourites.push(homeId);
        fs.writeFile(favouritesPath, JSON.stringify(favourites), (err) => {
          | console.log("File Writing Concluded", err);
        });
      }
    });
  }

  static getFavourites(callback) {
    fs.readFile(favouritesPath, (err, data) => {
      | callback(!err ? JSON.parse(data) : []);
    });
  }
};
```



14.5 Adding Favourites Model

2.

```
favourities.js  storeController.js  favourites.json
controllers > storeController.js > ...
54
55 exports.postFavourites = (req, res, next) => {
56   const homeId = req.body.homeId;
57   Favourites.addFavourite(homeId);
58   res.redirect('/favourites');
59 };
```

```
favourities.js  storeController.js  favourites.json
data > favourites.json > ...
1  ["1", "2"]|
```



14.5 Adding Favourites Model

3.

```
exports.getFavouriteList = (req, res, next) => {
  Favourites.getFavourites((favourites) => {
    Home.fetchAll((registeredHomes) => {
      const favouritesWithDetails = favourites.map(homeId => registeredHomes.find(home => home.id === homeId));
      res.render("store/favourite-list", {
        favourites: favouritesWithDetails,
        pageTitle: "My Favourites",
        currentPage: "favourites",
      });
    });
  });
};
```



14.5 Adding Favourites Model

4.

```
favourites.js  storeController.js  homes.json  favourites.json  <-- favourite-list.ejs  <-- home-detail.ejs  <-- home-list.ejs
views > store > <-- favourite-list.ejs > <-- >
1  <-- include('../partials/head') ><
2  </head>
3  <body>
4    <-- include('../partials/nav') >
5    <main class="container mx-auto bg-white shadow-lg rounded-lg p-8 mt-10 max-w-6xl">
6      <h2 class="text-3xl text-red-500 font-bold text-center mb-6">
7        Here are your favourites
8      </h2>
9      <div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">
10        <% favourites.forEach(home => { %>
11          <div class="bg-white rounded-lg shadow-md overflow-hidden hover:shadow-xl transition duration-300">
12            <div class="aspect-square">
13              " class="w-full h-full object-cover">
14            </div>
15            <div class="p-6">
16              <h3 class="text-2xl font-semibold text-gray-800 mb-3"><%= home.houseName %></h3>
17              <p class="text-gray-600 mb-4 flex items-center">
18                <svg xmlns="http://www.w3.org/2000/svg" class="h-5 w-5 mr-2" viewBox="0 0 20 20" fill="currentColor">
19                  <path fill-rule="evenodd" d="M5.05 4.05a7 7 0 119.9 9.9L10 18.9l-4.95-4.95a7 7 0 010-9.9zM10 11a2 2 0 100-4 2 2 0 000 4z" clip-rule="evenodd" />
20                </svg>
21                <%= home.location %>
22              </p>
23              <div class="flex justify-between items-center mb-4">
24                <span class="text-xl font-bold text-red-500"><%= home.price %> / night</span>
25                <div class="flex items-center bg-yellow-50 px-3 py-1 rounded-full">
26                  <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="currentColor" class="w-5 h-5 text-yellow-400 mr-1">
27                    <path fill-rule="evenodd" d="M10.788 3.21c.448-1.077 1.976-1.077 2.424 0l.082 5.007 5.404.433c1.164.093 1.636 1.545.749 2.305l-4.117 3.527 1.257 5.273c.271 1.136-.964 2.033-1.96 1.425L12 18.354 7.373 21.18c-.996 .608-2.231-.291-1.96-1.425l1.257-.527c-3.527-.887-.76-.415-2.212.749-2.305l5.404-.433 2.082-5.006z" clip-rule="evenodd" />
28                </svg>
29                <span class="text-gray-700 font-medium"><%= home.rating %></span>
30              </div>
31            </div>
32            <div class="flex gap-3">
33              <a href="/homes/<%= home.id %>" class="flex-1 bg-white text-red-500 py-2 px-4 rounded-lg text-center hover:bg-red-500 hover:text-white border border-red-500">View Details</a>
34            </div>
35          </div>
36        </div>
37        <% } %>
38      </div>
39    </main>
40  </body>
41 </html>
```



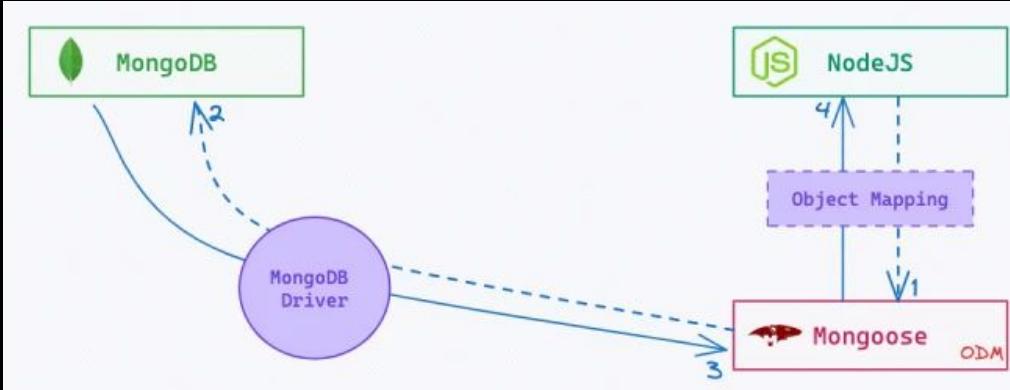
17. Introduction to Mongoose

1. What is Mongoose
2. Setting up Mongoose
3. Create Home Schema
4. Saving Home using Mongoose
5. Fetching Homes
6. Fetching one Home
7. Editing a Home
8. Deleting a Home
9. Using Mongoose for Favourite
10. Fetching Relations





17.1 What is Mongoose



1. Mongoose is an **Object Data Modeling (ODM)** library for **MongoDB** and **Node.js**.
2. Provides a **schema-based solution** to model application data.
3. Simplifies **data validation** and **type casting** in **Node.js** applications.
4. Enables easy interaction with **MongoDB** through intuitive methods.
5. Supports **middleware** for pre and post-processing of data.
6. Helps manage relationships between data with built-in functions.



17.2 Setting up Mongoose

mongoose

elegant **mongodb** object modeling for node.js

[read the docs](#)[discover plugins](#)

Version 8.8.0



Let's face it, **writing MongoDB validation, casting and business logic boilerplate is a drag**. That's why we wrote Mongoose.

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://127.0.0.1:27017/test');

const Cat = mongoose.model('Cat', { name: String });

const kitty = new Cat({ name: 'Zildjian' });
kitty.save().then(() => console.log('meow'));
```

Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

[Get Professionally Supported Mongoose](#)



17.2 Setting up Mongoose

1. Install **Mongoose** package.
2. Import and use **mongoose** in **app.js**
3. Delete the **database-util** file.
4. Remove the **usage of db-util** from everywhere.

1.

```
● prashantjain@Prashants-Mac-mini:12 ~ % npm install mongoose
added 8 packages, and audited 258 packages in 1s
48 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```



17.2 Setting up Mongoose

2.

```
const PORT = 3001;
mongoose.connect("mongodb+srv://root:root@kgcluster.ie6mb.mongodb.net/airbnb?
retryWrites=true&w=majority&appName=KGCluster").then(() => {
  console.log("Connected to MongoDB");
  app.listen(PORT, () => {
    console.log(`Server running at: http://localhost:${PORT}`);
  });
}).catch((err) => {
  console.log("Error while connecting to MongoDB", err);
});
```



17.3 Create Home Schema

1. Delete complete **airbnb** db from mongo.
2. Delete the existing **Home Model** code.
3. Create the new **Home Schema** in the **Home Model File**.

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

// _id is automatically added by mongoose
const homeSchema = new mongoose.Schema({
  houseName: {type: String, required: true},
  price: {type: Number, required: true},
  location: {type: String, required: true},
  rating: {type: Number, required: true},
  photoUrl: String,
  description: String
});
```



17.4 Saving Home using Mongoose

```
module.exports = mongoose.model("Home", homeSchema);

exports.postAddHome = (req, res, next) => {
  const { houseName, price, location, rating, photoUrl, description } =
    req.body;
  const newHome = new Home({
    houseName,
    price,
    location,
    rating,
    photoUrl,
    description
  });

  newHome.save().then((rows) => {
    res.render("host/home-added", { pageTitle: "Home Hosted" });
  });
};
```



17.5 Fetching Homes

The screenshot shows the VS Code interface with a search bar at the top containing the query "fetchAll". Below the search bar is a list of results from four files. The results are as follows:

- hostController.js: Home.fetchAll().find().then((registeredHomes) => {
- storeController.js: Favourite.fetchAll().find().then((favouritelds) => {
- Favourite.js: static fetchAll() find() {
- Home.js: // static fetchAll() find() {

Each result has a small blue circular badge with the number "1" next to it, indicating there is one occurrence of the search term in each file.

```
exports.getIndex = (req, res, next) => {
  Home.find().then((registeredHomes) => {
    res.render("store/index", {
      homes: registeredHomes,
      pageTitle: "Tumahara airbnb",
    });
  });
};
```



17.6 Fetching one Home

It already works !!!



17.7 Editing a Home

```
exports.postEditHome = (req, res, next) => {
  const { id, houseName, price, location, rating, photoUrl, description } = req.body;
  Home.findById(id).then((home) => {
    if (!home) {
      console.log("Home not found for editing");
      return res.redirect("/host/host-homes");
    }
    home.houseName = houseName;
    home.price = price;
    home.location = location;
    home.rating = rating;
    home.photoUrl = photoUrl;
    home.description = description;
    return home.save();
  }).then(() => {
    res.redirect("/host/host-homes");
  })
  .catch((err) => {
    console.log("Error while updating home", err);
  });
};
```



17.8 Deleting a Home

```
exports.postDeleteHome = (req, res, next) => {
  const homeId = req.params.homeId;
  console.log("Came to delete ", homeId);
  Home.findByIdAndDelete(homeId).then(() => {
    res.redirect("/host/host-homes");
  });
};
```



17.9 Using Mongoose for Favourite

1. Delete the existing Favourite Model code.
2. Create the new Favourite Schema in the Favourite Model File.
3. Fix the following functionalities:
 - a. Getting all Favourite
 - b. Adding A Favourite
 - c. Deleting a Favourite
4. Removing Favourite while removing home.



17.9 Using Mongoose for Favourite

1, 2.

```
const mongoose = require('mongoose');
...
const favouriteSchema = new mongoose.Schema({
  homeId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Home',
    required: true,
    unique: true
  }
});
module.exports = mongoose.model('Favourite', favouriteSchema);
```



17.9 Using Mongoose for Favourite

```
3.a. exports.getFavourites = (req, res, next) => {
  Favourite.find().then((favouriteIds) => {
    favouriteIds = favouriteIds.map((favourite) => favourite.homeId.toString());
    Home.find().then((registeredHomes) => {
      console.log(favouriteIds);
      console.log(registeredHomes);
      const favouriteHomes = registeredHomes.filter((home) =>
        favouriteIds.includes(home._id.toString())
      );
      res.render("store/favourites", {
        homes: favouriteHomes,
        pageTitle: "Favourites",
      });
    });
  });
};
```



17.9 Using Mongoose for Favourite

```
3.b. exports.postAddFavourites = (req, res, next) => {
  const homeId = req.body.id;
  Favourite.findOne({ homeId: homeId })
    .then(existingFav => {
      if (existingFav) {
        return res.redirect("/favourites");
      }
      const fav = new Favourite({ homeId: homeId });
      return fav.save();
    })
    .then(() => {
      res.redirect("/favourites");
    })
    .catch((err) => {
      console.log("Error while adding to favourites", err);
    });
};
```



17.9 Using Mongoose for Favourite

3.c.

```
exports.postRemoveFavourite = (req, res, next) => {
  const homeId = req.params.homeId;
  Favourite.findOneAndDelete({ homeId: homeId })
    .then(() => {
      res.redirect("/favourites");
    })
    .catch((err) => {
      console.log("Error while remove from favourites ", err);
    });
};
```



17.9 Using Mongoose for Favourite

4.

```
homeSchema.pre('findOneAndDelete', async function(next) {
  const homeId = this.getQuery()['_id'];
  await Favourite.deleteMany({ homeId: homeId });
  next();
});
```



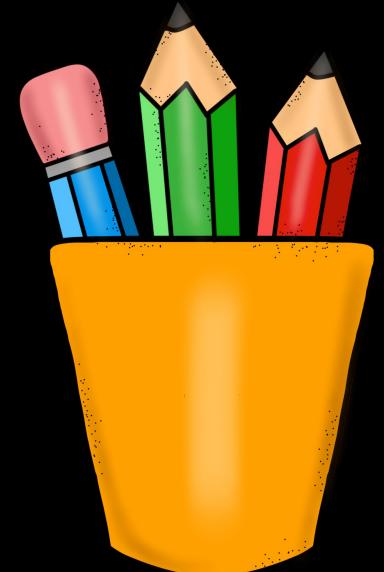
17.10 Fetching Relations

```
exports.getFavourites = (req, res, next) => {
  Favourite.find()
    .populate("homeId")
    .then((favourites) => {
      const favouriteHomes = favourites.map((favourite) => favourite.homeId);
      res.render("store/favourites", {
        homes: favouriteHomes,
        pageTitle: "Favourites",
      });
    })
};
```



Revision

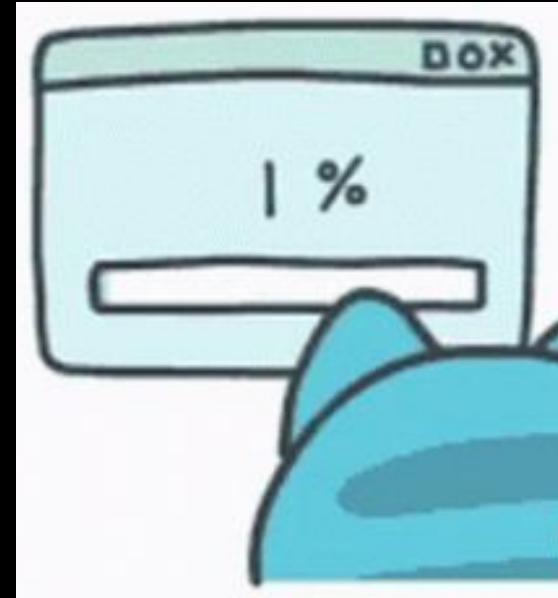
1. What is Mongoose
2. Setting up Mongoose
3. Create Home Schema
4. Saving Home using Mongoose
5. Fetching Homes
6. Fetching one Home
7. Editing a Home
8. Deleting a Home
9. Using Mongoose for Favourite
10. Fetching Relations





21. File Upload & Download

1. Adding a File Picker
2. Creating Multipart Form
3. Handling Multipart Form Data
4. Saving Image Files
5. Custom File Names
6. Restricting Upload File Types
7. Handling Edits
8. Serving Saved Data
9. Serving Files after Auth
10. Deleting Files





21.1 Adding a File Picker

- **Input Element:** Use `<input type="file">` to create a file picker.
- **Multiple Files:** Add `multiple` attribute to allow selecting multiple files.
- **File Types:** Use `accept` attribute to restrict file types (e.g., `accept=".jpg, .png"`).

```
<input type="file"  
      name="photo"  
      class="w-full px-4 py-2 mb-4 border border-gray-300  
      rounded-lg focus:outline-none focus:border-blue-500"  
/>
```

Add your Home

Name of your house

Daily rent of your home

Where is your home

Rating of the house

No file chosen

Describe your house



21.2 Creating Multipart Form

```
exports.postAddHome = (req, res, next) => {
  const { houseName, price, location, rating, photoUrl, description } = req.body;
  console.log(req.body);
```

```
Server running at: http://localhost:3001
{
  id: '',
  houseName: 'New House',
  price: '1299',
  location: 'kashmir',
  rating: '3.9',
  photo: 'IMG_4705.HEIC',
  description: 'asdfasdfs'
}
```

| X | Headers | Payload | Preview | Response | Initiator | Timing | Cookies |
|------------------------|---------|---|---------|----------|-----------|--------|---------|
| ▼ General | | | | | | | |
| Request URL: | | http://localhost:3001/host/edit-home | | | | | |
| Request Method: | | POST | | | | | |
| Status Code: | | ● 302 Found | | | | | |
| Remote Address: | | [::1]:3001 | | | | | |
| Referrer Policy: | | strict-origin-when-cross-origin | | | | | |
| ► Response Headers (8) | | | | | | | |
| ▼ Request Headers | | Raw | | | | | |
| Accept: | | text/html,application/xhtml+xml,application/xml;q=0.9,image | | | | | |
| Accept-Encoding: | | gzip, deflate, br, zstd | | | | | |
| Accept-Language: | | en-GB,en;q=0.9 | | | | | |
| Cache-Control: | | max-age=0 | | | | | |
| Connection: | | keep-alive | | | | | |
| Content-Length: | | 114 | | | | | |
| Content-Type: | | application/x-www-form-urlencoded | | | | | |
| Cookie: | | connect.sid=s%3AJHZ1M5GAnP9QuJSwuUTfVrbHSq6HBC7 | | | | | |
| Host: | | localhost:3001 | | | | | |
| Origin: | | http://localhost:3001 | | | | | |



21.3 Handling Multipart Form Data

prashantjain@Prashants-MacBook-Pro:~\$ email and advance auth % `npm install multer`

added 17 packages, and audited 298 packages in 719ms

50 packages are looking for funding
run `npm fund` for details

1 **high** severity vulnerability

To address all issues, run:
`npm audit fix`

Run `npm audit` for details.

multer DT

1.4.5-lts.1 • Public • Published 2 years ago

[Readme](#) [Code](#) Beta [7 Dependencies](#) [4,749 Dependents](#) [46 Versions](#)

Multer build unknown [npm package](#) [1.4.5-lts.1](#) [code style standard](#)

Multer is a node.js middleware for handling `multipart/form-data`, which is primarily used for uploading files. It is written on top of `busboy` for maximum efficiency.

NOTE: Multer will not process any form which is not multipart (`multipart/form-data`).

Translations

This README is also available in other languages:

- [Español](#) (Spanish)
- [简体中文](#) (Chinese)
- [한국어](#) (Korean)
- [Русский язык](#) (Russian)
- [Việt Nam](#) (Vietnam)
- [Português](#) (Portuguese Brazil)

Installation

```
$ npm install --save multer
```

Install

```
> npm i multer
```

Repository

github.com/expressjs/multer

Homepage

github.com/expressjs/multer#readme

Weekly Downloads

6,313,138



Version

1.4.5-lts.1

License

MIT

Unpacked Size

27.6 kB

Total Files

11

Issues

[195](#)

Pull Requests

[70](#)



21.3 Handling Multipart Form Data

```
<form action="/host/<%= editing ? "edit-home" : "add-home"%>" method="POST" enctype="multipart/form-data"
class="w-full max-w-md">
```

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

app.use(express.static(path.join(rootDir, "public")));
app.use(bodyParser.urlencoded({ extended: true }));
```

```
console.log(req.file);
```

```
{
 fieldname: 'photo',
originalname: 'IMG_4705.HEIC',
encoding: '7bit',
mimetype: 'image/heic',
buffer: <Buffer 00 00 00 1c 66 74 79 70 68 65 69 63 00 00 00 00 74 6d
00 00 21 68 64 6c 72 00 00 ... 2246669 more bytes>,
size: 2246719
}
```



21.4 Saving Image Files

```
app.use(multer({ dest: 'uploads/' }).single('photo'));
```

```
{
 fieldname: 'photo',
originalname: 'house1.png',
encoding: '7bit',
mimetype: 'image/png',
destination: 'uploads/',
filename: '0c710a9b2903e323fc38e7926650d159',
path: 'uploads/0c710a9b2903e323fc38e7926650d159',
size: 387102
}
```

```
> 🌐 routers
└─ uploads
    └─ 0c710a9b2903e323fc38e7926650d159
└─ util
```



21.5 Custom File Names

```
// This defines where uploaded files will be stored and how they'll be named
const storage = multer.diskStorage({
  // Set the destination folder for uploaded files
  destination: (req, file, cb) => {
    | cb(null, 'uploads/'); // Files will be saved in the 'uploads' directory
  },
  // Set the filename for uploaded files
  filename: (req, file, cb) => {
    | cb(null, new Date().toISOString() + '-' + file.originalname);
  }
});
```

```
app.use(multer({ storage }).single('photo'));
```

```
> 🌱 routers
└─ uploads
  └─ 0c710a9b2903e323fc38e7926650d159
    └─ 2024-11-25T11:26:17.582Z-house1.png
└─ util
```



21.6 Restricting Upload File Types

```
const fileFilter = (req, file, cb) => {
  if ([ 'image/jpeg', 'image/png', 'image/jpg' ].includes(file.mimetype)) {
    cb(null, true);
  } else {
    cb(null, false);
  }
};
```

```
app.use(multer({ storage, fileFilter }).single('photo'));
```

Adding this filter, now multer would not have the file if it does not match the type and req.file will just be undefined.

```
exports.postEditHome = (req, res, next) => {
  const { id, houseName, price, location, rating, photoUrl, description } =
    req.body;

  if (!req.file) {
    return res.status(400).send('No image provided');
  }
```



21.6 Restricting Upload File Types

```
if (!req.file) {  
  return res.status(400).send('No image provided');  
}  
  
const photoUrl = req.file.path;
```

```
_id: ObjectId('67445aa6e25a19de45a1b82e')  
houseName : "New House"  
price : 1299  
location : "kashmir"  
rating : 3.9  
description : "asdfasdf"  
host : ObjectId('673cacd23e8557da8e22d9b1')  
__v : 0  
photoUrl : "uploads/2024-11-25T11:35:47.086Z-house1.png"
```



21.7 Handling Edits

```
// business logic outside model
Home.findById(id)
  .then((existingHome) => {
    if (!existingHome) {
      console.log("Home not found for editing");
      return res.redirect("/host/host-homes");
    }
    existingHome.houseName = houseName;
    existingHome.price = price;
    existingHome.location = location;
    existingHome.rating = rating;
    if (req.file) {
      existingHome.photoUrl = req.file.path;
    }
    existingHome.description = description;
    return existingHome.save();
  })
  .catch((err) => {
    console.error(err);
    res.status(500).send("Internal Server Error");
  });
}

module.exports = router;
```

While editing we can make sure that we only overwrite the photoUrl in case a new valid image was uploaded. And don't force the user to upload images each time they edit.



21.8 Serving Saved Data

```
app.use(express.static(path.join(rootDir, "public")));
app.use('/uploads', express.static(path.join(rootDir, "uploads")));
app.use(bodyParser.urlencoded({ extended: true }));
```

This is done as like in case of public, things inside public will be served like they are in root folder and the url does not have public in it. Here also either remove uploads/ from the image path, or add the qualifier.



21.9 Serving Files after Auth



Property Details

Description

Experience luxury living in this beautiful property. This home offers the perfect blend of comfort and style, situated in the prime location of goa. With its exceptional rating of 4.5/5, this property stands out as one of our finest offerings.

[Download House Rules](#)

[Add to Favourite](#)

Features

- ✓ Prime Location
- ✓ Highly Rated
- ✓ Modern Amenities

```
<div class="mt-4">
  <a href="/rules/<%= home._id %>" class="inline-flex items-center text-blue-600 hover:text-blue-800">
    <svg class="w-5 h-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">
      <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 10v6m0 0l-3-3m3 3l3-3m2 8H7a2 2 0 01-2-2V5a2 2 0 012-2h5.586a1 1 0 01.707.293l5.414 5.414a1 1 0 01.293.707V19a2 2 0 01-2 2z"/>
    </svg>
    Download House Rules
  </a>
</div>
```



21.9 Serving Files after Auth

```
storeRouter.get("/rules/:homeId", storeController.getHouseRules);

const path = require('path');
const rootDir = require('../util/path-util');

exports.getHouseRules = [(req, res, next) => {
  if (!req.session.isLoggedIn) {
    return res.redirect("/login");
  }
  next();
},
(req, res, next) => {
  const homeId = req.params.homeId;
  // We can make it house specific after have homeId in file name like this: `House Rules-${homeId}.pdf`
  const rulesFileName = `House Rules.pdf`;
  const filePath = path.join(rootDir, 'rules', rulesFileName);
  // res.sendFile(filePath);
  res.download(filePath, 'Rules.pdf');
}
];
```

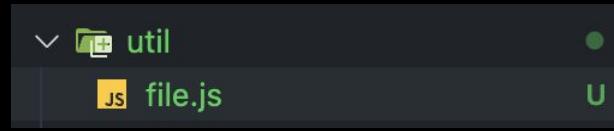


21.10 Deleting Files

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

exports.deleteFile = (filePath) => {
  fs.unlink(filePath, (err) => {
    if (err) throw err;
  });
};
```

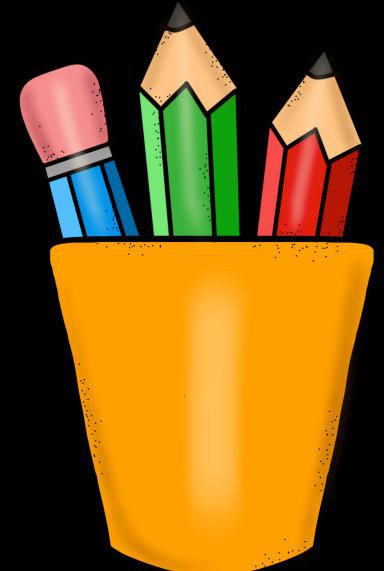
```
existingHome.location = location;
existingHome.rating = rating;
if (req.file) {
  deleteFile(existingHome.photoUrl);
  existingHome.photoUrl = req.file.path;
}
```





Revision

1. Adding a File Picker
2. Creating Multipart Form
3. Handling Multipart Form Data
4. Saving Image Files
5. Custom File Names
6. Restricting Upload File Types
7. Handling Edits
8. Serving Saved Data
9. Serving Files after Auth
10. Deleting Files





Practise Milestone

Take your **airbnb** forward:

1. Add a **new button** for **rules upload** on the **edit page**.
2. Make it **downloadable** on the **home detail page**.

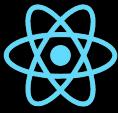




React

5.6 Parsing Request

```
req.on("end", () => {
  const parsedBody = Buffer.concat(body).toString();
  console.log(parsedBody);
  const params = new URLSearchParams(parsedBody);
  const jsonObject = {};
  for (const [key, value] of params.entries()) {
    jsonObject[key] = value;
  }
  console.log(jsonObject);
  // Output: { name: 'Prashant', gender: 'male' }
});
fs.writeFileSync("user-details.txt", "Prashant Jain");
res.setHeader("Location", "/");
res.statusCode = 302;
return res.end();
```



React

5.6 Parsing Request

```
req.on("end", () => {
  const parsedBody = Buffer.concat(body).toString();
  console.log(parsedBody);
  const params = new URLSearchParams(parsedBody);
  const jsonObject = {};
  for (const [key, value] of params.entries()) {
    jsonObject[key] = value;
  }
  const jsonString = JSON.stringify(jsonObject);
  console.log(jsonString);
  fs.writeFileSync("user-details.txt", jsonString);
});

res.setHeader("Location", "/");
res.statusCode = 302;
return res.end();
```

```
node > ≡ user-details.txt
```

```
1  {"name":"Prashant","gender":"male"}
```



React

JS app.js

JS handler.js

```
JS handler.js > ...
const fs = require("fs");

const requestHandler = (req, res) => {
  if (req.url === "/") {
    res.setHeader("Content-Type", "text/html");

module.exports = requestHandler
```

JS app.js > ...

```
// Simple NodeJS server
const http = ...
const requestHandler = require('./handler');

const server = http.createServer(requestHandler);
```



5.7 Using Modules

```
// Method 1: Multiple exports using object
module.exports = {
  handler: requestHandler,
  extra: "Extra"
};

// Method 2: Setting multiple properties
module.exports.handler = requestHandler;
module.exports.extra = "Extra";

// Method 3: Shortcut using exports
exports.handler = requestHandler;
exports.extra = "Extra";
```



React

Practise Set

Create a Calculator

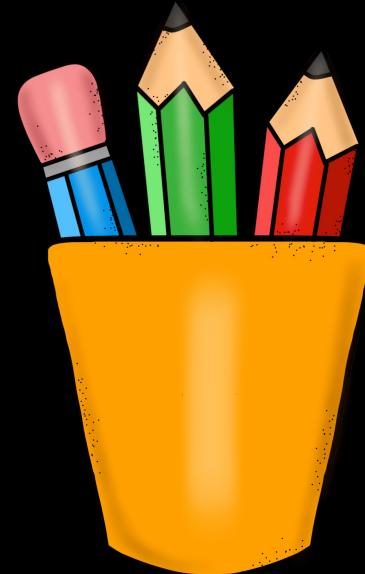
1. Create a new Node.js project named “Calculator”.
2. On the home page (route “/”), show a welcome message and a link to the calculator page.
3. On the “/calculator” page, display a form with two input fields and a “Sum” button.
4. When the user clicks the “Sum” button, they should be taken to the “/calculate-result” page, which shows the sum of the two numbers.
 - o Make sure the request goes to the server.
 - o Create a separate module for the addition function.
 - o Create another module to handle incoming requests.
 - o On the “/calculate-result” page, parse the user input, use the addition module to calculate the sum, and display the result on a new HTML page.

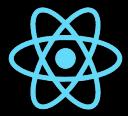




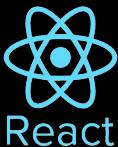
Revision

1. Streams
2. Chunks
3. Buffers
4. Reading Chunk
5. Buffering Chunks
6. Parsing Request
7. Using Modules





React



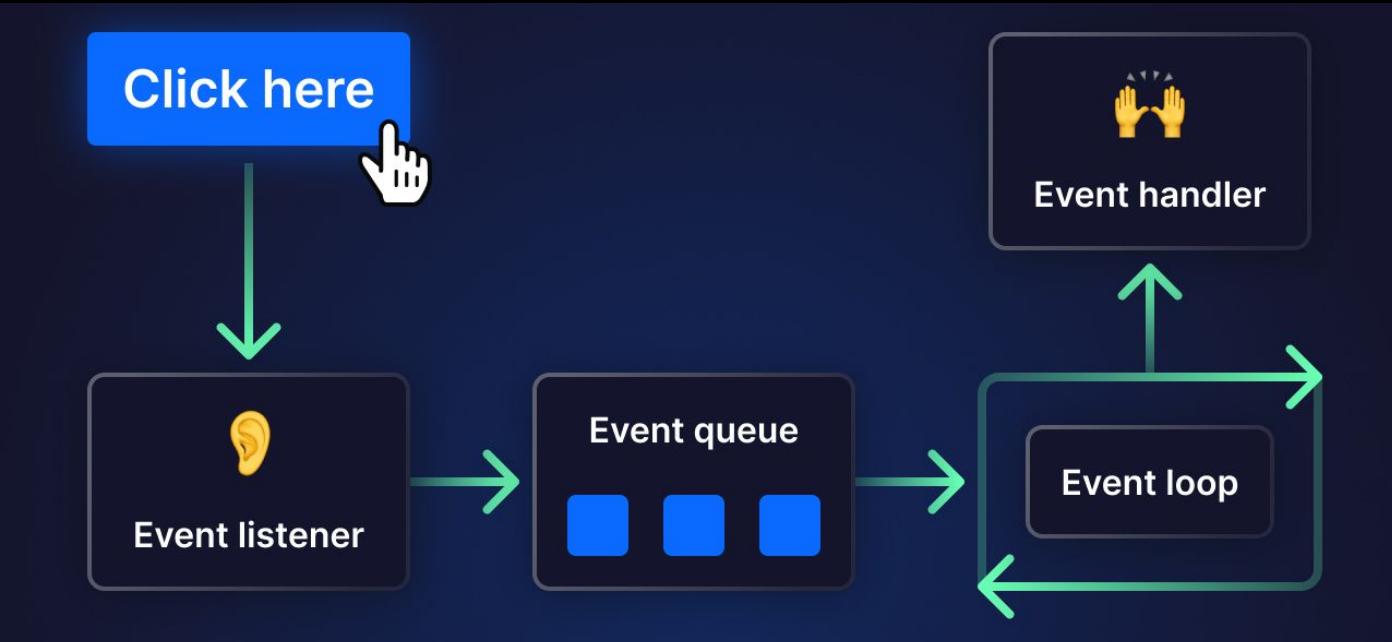
6. Event Loop

1. Event Driven
2. Single Threaded
3. V8 vs libuv
4. Node Runtime
5. Event Loop
6. Async Code
7. Blocking Code



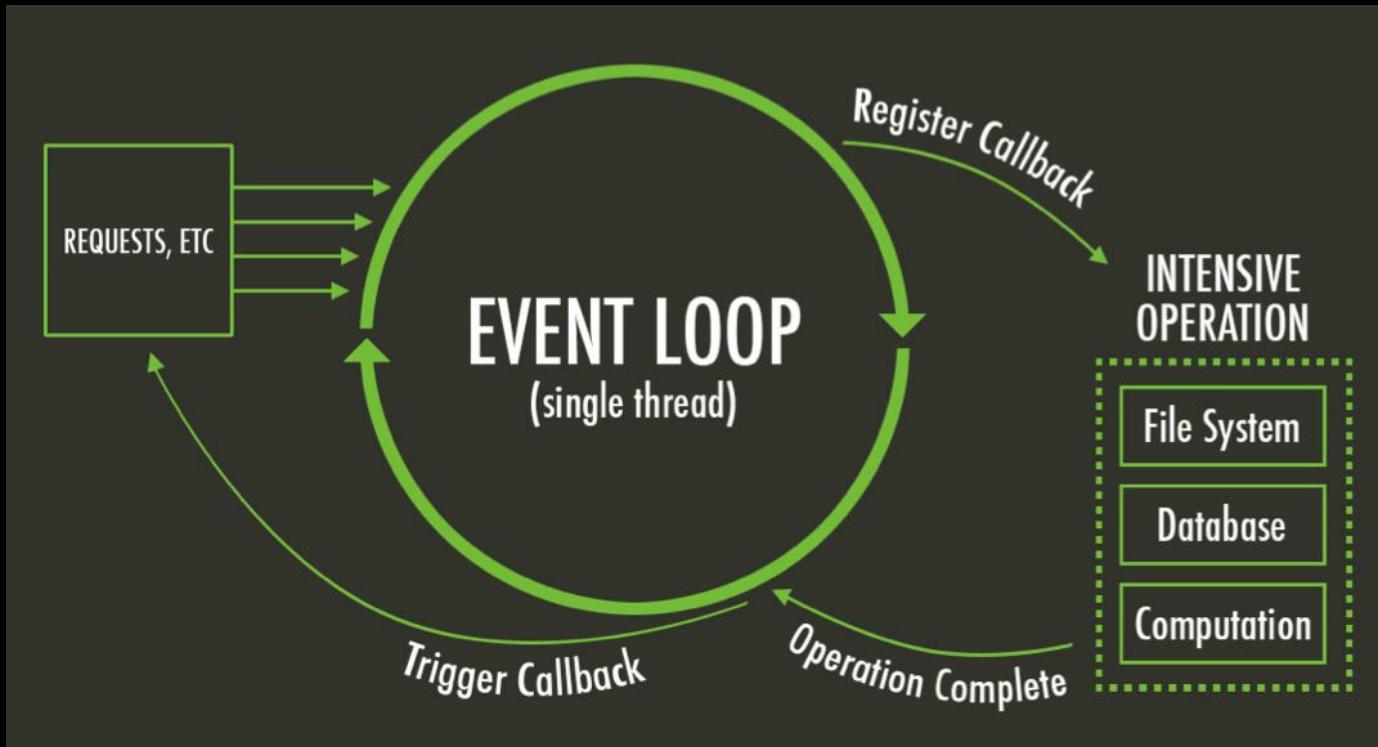


6.1 Event Driven





6.2 Single Threaded





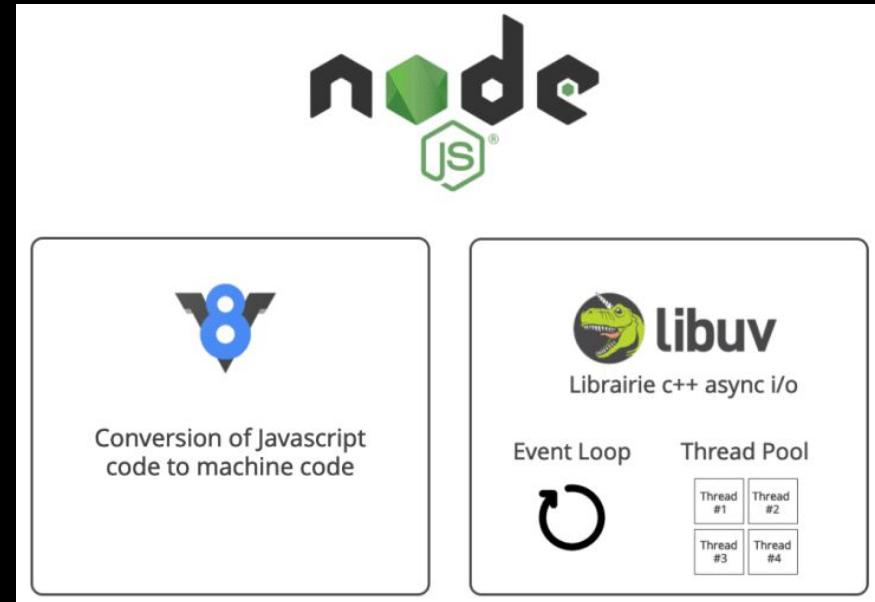
6.3 V8 vs libuv

V8:

1. Open-source JavaScript engine by Google.
2. Used in Chrome and Node.js.
3. Compiles JavaScript to native machine code.
4. Ensures high-performance JavaScript execution.

libuv:

1. Multi-platform support library for Node.js.
2. Handles asynchronous I/O operations.
3. Provides event-driven architecture.
4. Manages file system, networking, and timers non-blockingly across platforms.



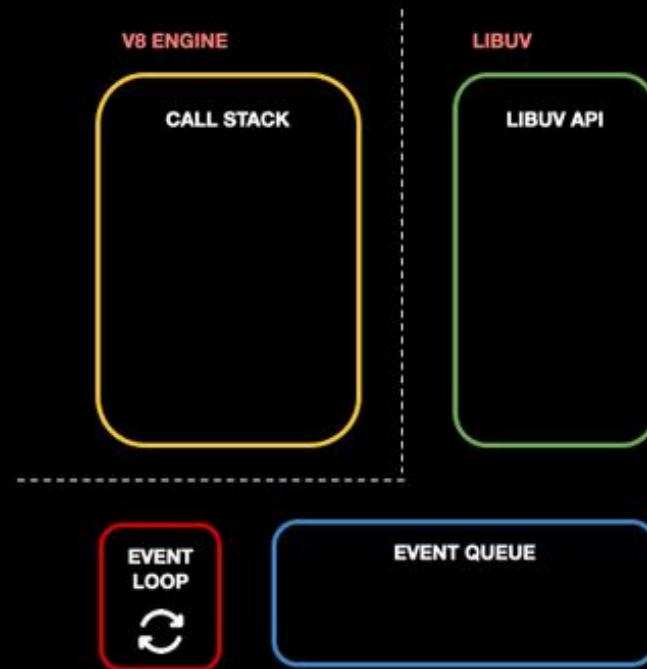


6.4 Node Runtime

An invoked function is added to the call stack. Once it returns a value, it is popped off.

```
● ● ●  
console.log("Starting Node.js");  
  
db.query("SELECT * FROM public.cars", function (err, res) {  
  console.log("Query executed");  
});  
  
console.log("Before query result");
```

OUTPUT





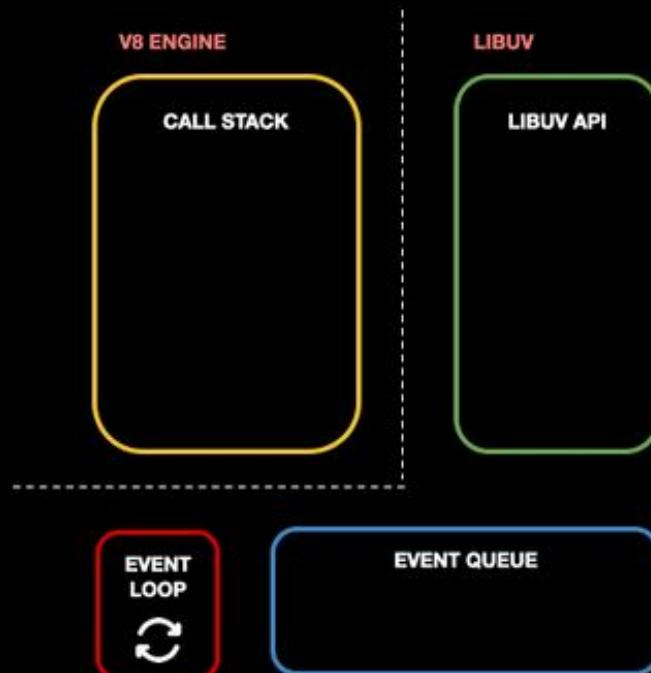
6.4 Node Runtime

Database queries or other I/O ops do not block Node.js single thread because Libuv API handles them.

```
→ console.log("Starting Node.js");

db.query("SELECT * FROM public.cars", function (err, res) {
  console.log("Query executed");
});

console.log("Before query result");
```



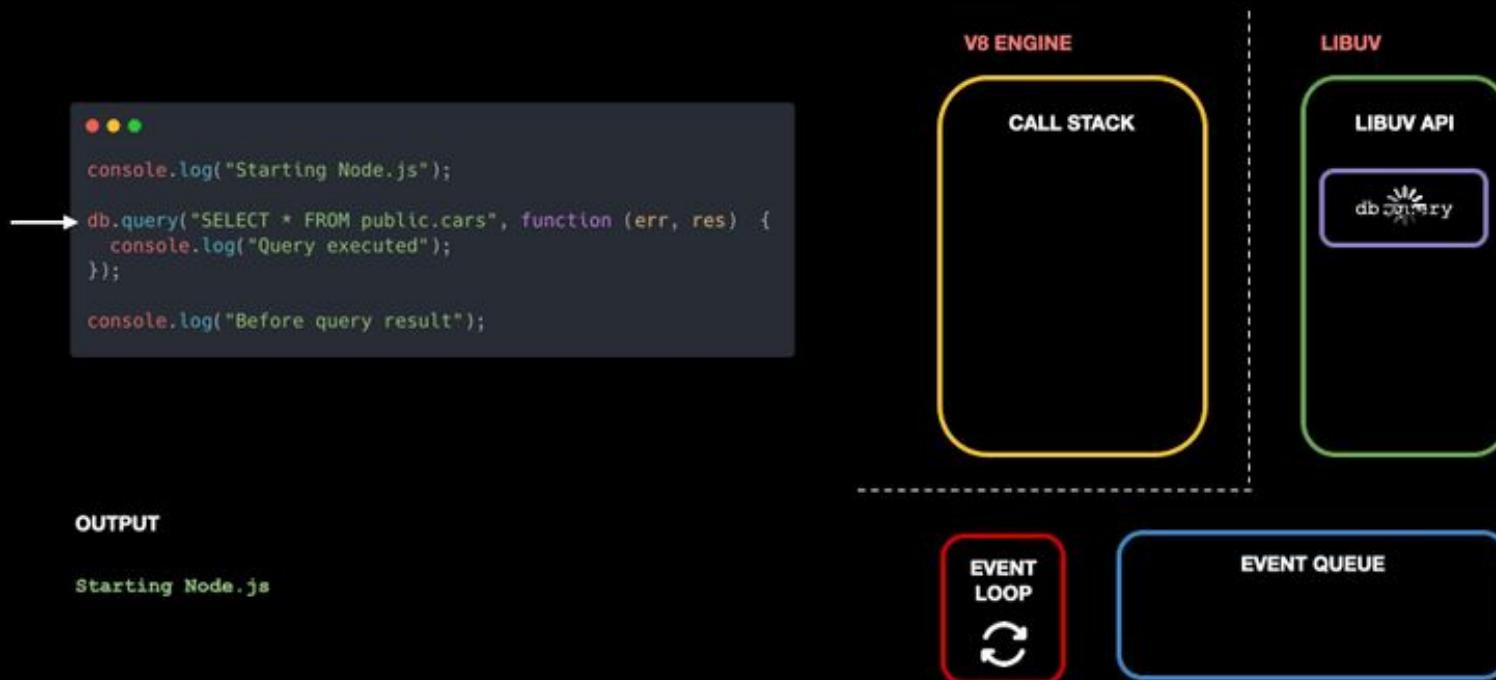
OUTPUT

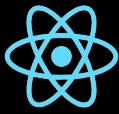
Starting Node.js



6.4 Node Runtime

While Libuv asynchronously handles I/O operations, Node.js single thread keeps running code.



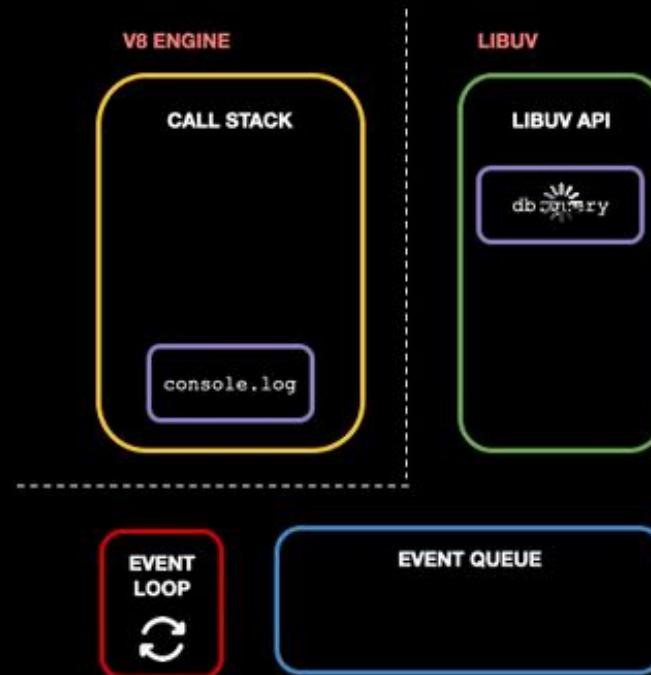


React

6.4 Node Runtime

Callbacks of completed queries are moved to the event queue. If the call stack is empty, the event loop checks for callbacks and transfers the first.

```
● ● ●  
console.log("Starting Node.js");  
  
db.query("SELECT * FROM public.cars", function (err, res) {  
  console.log("Query executed");  
});  
  
→ console.log("Before query result");
```



OUTPUT

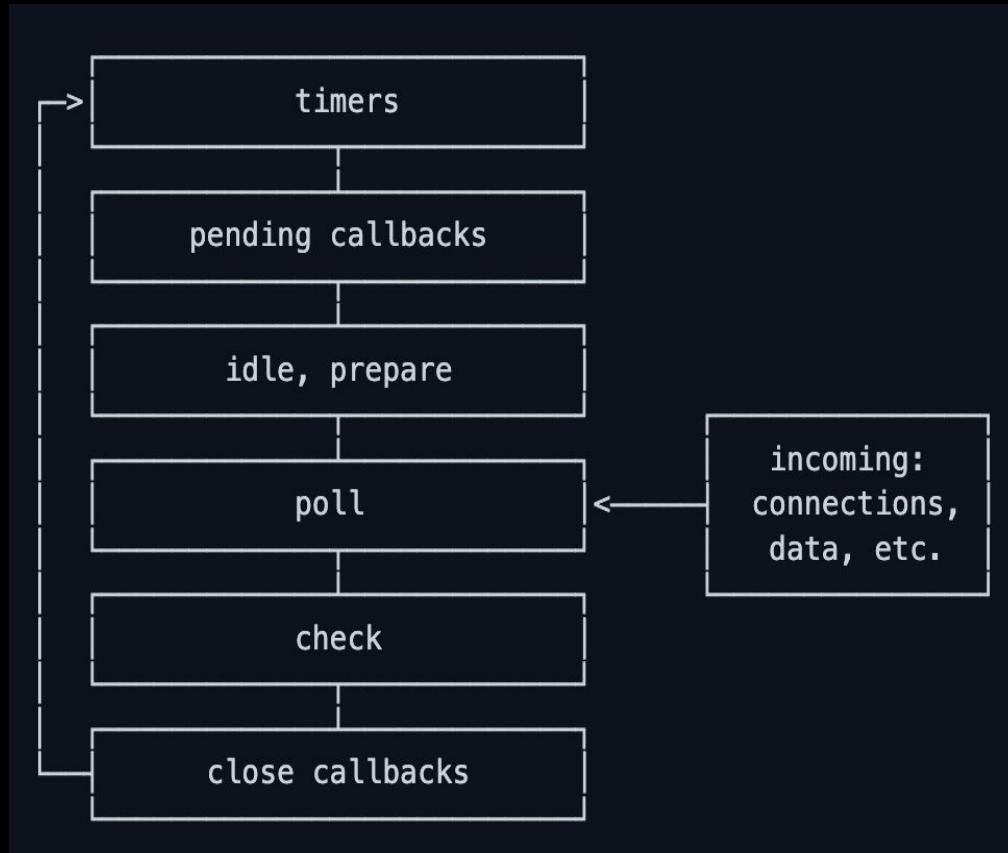
Starting Node.js

Before query result



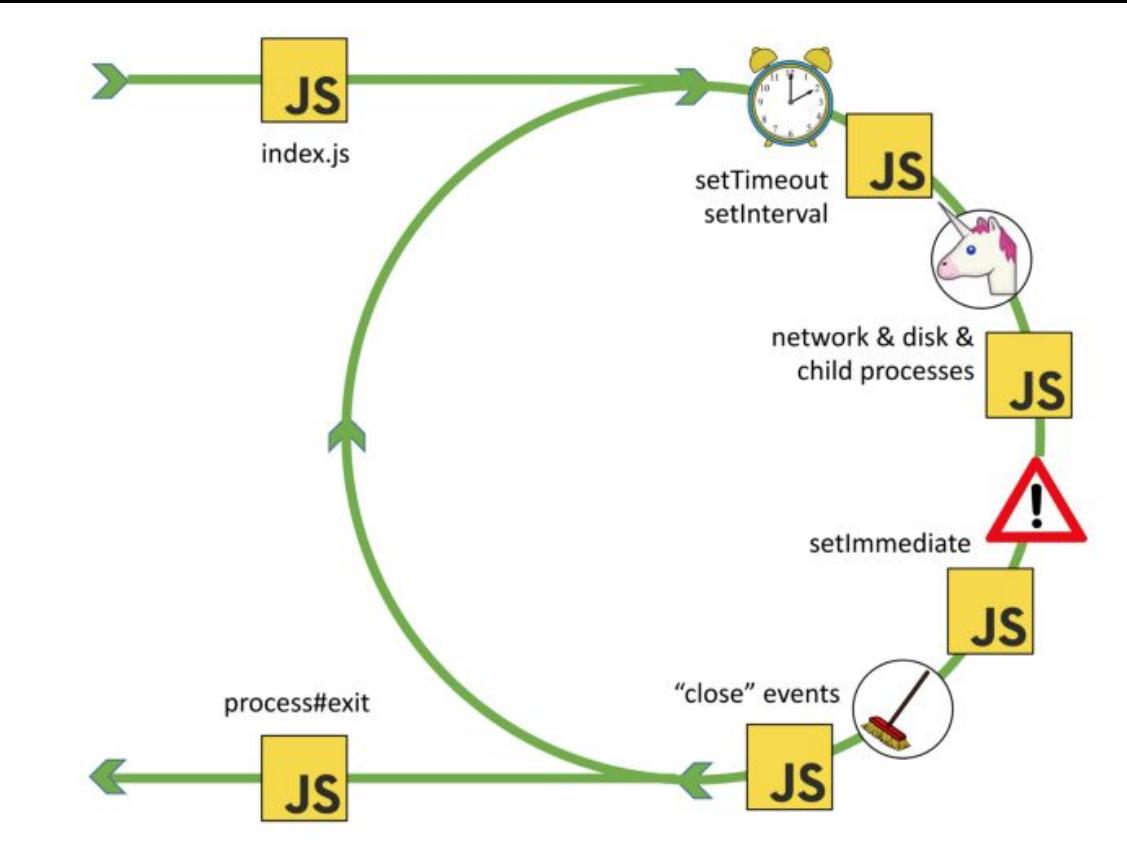
6.5 Event Loop

- **timers**: this phase executes callbacks scheduled by `setTimeout()` and `setInterval()`.
- **pending callbacks**: executes I/O callbacks deferred to the next loop iteration.
- **idle, prepare**: only used internally.
- **poll**: retrieve new I/O events; execute I/O related callbacks (almost all with the exception of close callbacks, the ones scheduled by **timers**, and `setImmediate()`); node will block here when appropriate.
- **check**: `setImmediate()` callbacks are invoked here.
- **close callbacks**: some close callbacks, e.g. `socket.on('close', ...)`.





6.5 Event Loop





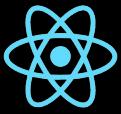
6.6 Async Code

```
const jsonString = JSON.stringify(jsonObject);
console.log(jsonString);
fs.writeFileSync("user-details.txt", jsonString);
res.setHeader("Location", "/");
res.statusCode = 302;
res.end();
});
```

```
}
```

```
res.write('<body><h1>Like / Share / Subscribe</h1></
body>');
res.write('</html>');
return res.end();
```

```
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
at ServerResponse.setHeader (node:_http_outgoing:699:11)
at IncomingMessage.<anonymous> (/Users/prashantjain/workspace/Test Project/node/app.js:44:11)
at IncomingMessage.emit (node:events:532:35)
at endReadableNT (node:internal/streams/readable:1696:12)
at process.processTicksAndRejections (node:internal/process/task_queues:82:21) {
  code: 'ERR_HTTP_HEADERS_SENT'
}
```



React

6.6 Async Code

```
req.on("end", () => {
  const parsedBody = Buffer.concat(body).toString();
  console.log(parsedBody);
  const params = new URLSearchParams(parsedBody);
  const jsonObject = {};
  for (const [key, value] of params.entries()) {
    jsonObject[key] = value;
  }
  const jsonString = JSON.stringify(jsonObject);
  console.log(jsonString);
  fs.writeFileSync("user-details.txt", jsonString);
  res.setHeader("Location", "/");
  res.statusCode = 302;
  return res.end();
});
```



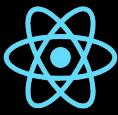
6.7 Blocking Code

```
const jsonString = JSON.stringify(jsonObject);
console.log(jsonString);
// BLOCKING EVERYTHING
fs.writeFileSync("user-details.txt", jsonString);
res.setHeader("Location", "/");
```



6.7 Blocking Code

```
console.log(jsonString);
// Async Operation
fs.writeFile("user-details.txt", jsonString, error => {
  res.setHeader("Location", "/");
  res.statusCode = 302;
  return res.end();
});
```



React

Run & Observe

Blocking vs Async

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

console.log('1. Start of script');

// Synchronous (blocking) operation
console.log('2. Reading file synchronously');
const dataSync = fs.readFileSync('user-details.txt', 'utf8');
console.log('3. Synchronous read complete');

// Asynchronous (non-blocking) operation
console.log('4. Reading file asynchronously');
fs.readFile('user-details.txt', 'utf8', (err, dataAsync) => {
  if (err) throw err;
  console.log('6. Asynchronous read complete');
});

console.log('5. End of script');
```



1. Start of script
2. Reading file synchronously
3. Synchronous read complete
4. Reading file asynchronously
5. End of script
6. Asynchronous read complete



Run & Observe

Event Loop Sequence

```
console.log('1. Start of script');

// Microtask queue (Promise)
Promise.resolve().then(() => console.log('2. Microtask 1'));

// Timer queue
setTimeout(() => console.log('3. Timer 1'), 0);

// I/O queue
const fs = require('fs');
fs.readFile('user-details.txt', () => console.log('4. I/O operation'));

// Check queue
setImmediate(() => console.log('5. Immediate 1'));

// Close queue
process.on('exit', (code) => {
  console.log('6. Exit event');
});

console.log('7. End of script');
```



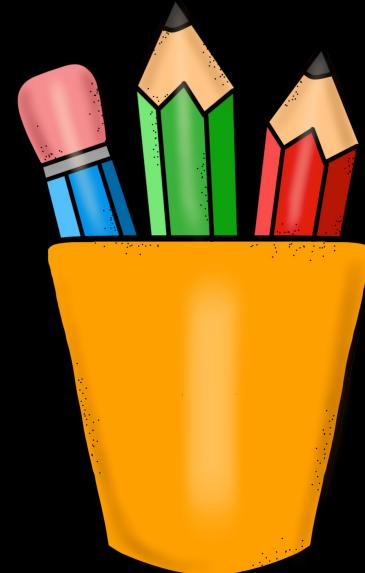
1. Start of script
2. Microtask 1
3. Timer 1
4. I/O operation
5. Immediate 1
6. Exit event
7. End of script



React

Revision

1. Event Driven
2. Single Threaded
3. V8 vs libuv
4. Node Runtime
5. Event Loop
6. Async Code
7. Blocking Code





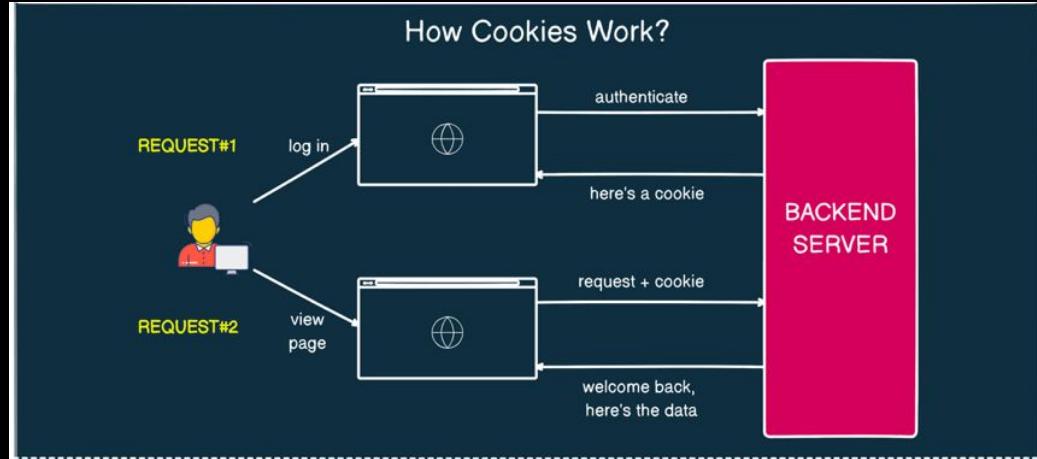
18. Cookies & Sessions

1. What are Cookies
2. Adding Login Functionality
3. Checking Login State
4. Using a Cookie
5. Define the Logout Feature
6. Problem with Cookies
7. What are Sessions
8. Installing Session Package
9. Creating a Sessions
10. Saving Session in DB





18.1 What are Cookies



1. Cookies are small pieces of data stored in the user's browser by server.
2. They help websites remember user information and preferences between page loads or visits.
3. Cookies can manage user sessions and store data for personalized experiences.



18.1 What are Cookies

Screenshot of the Chrome DevTools Network tab showing cookies for learn.completecoding.in.

| Name | Value | Domain | Path | Expires / Max-Age | Size | HttpOnly |
|--|---|-------------------------|------|--------------------------|------|----------|
| MUID | 0644EB8141A266AA049EFEB74050676E | .clarity.ms | / | 2025-12-08T09:00:21.314Z | 36 | |
| _clk | 1sddxv4%7C2%7Cfw%7C0%7C1778 | .completecoding.in | / | 2025-11-15T03:08:42.000Z | 33 | |
| _clsk | 10b84hl%7C1731668781029%7C2%7C0%7Ck.clarity.... | .completecoding.in | / | 2024-11-16T11:06:21.000Z | 61 | |
| _ga | GA1.1.585233942.1731488419 | .completecoding.in | / | 2025-12-20T03:08:42.237Z | 29 | |
| _ga_8JBLENZJ2Z | GS1.1.1731640107.6.1.1731640122.0.0.0 | .completecoding.in | / | 2025-12-20T03:08:42.171Z | 51 | |
| _ga_QBNBN7VB0P | GS1.1.1731640107.6.1.1731640122.0.0.0 | .completecoding.in | / | 2025-12-20T03:08:42.242Z | 51 | |
| _gcl_au | 1.1.1428947795.1731488419 | .completecoding.in | / | 2025-02-11T09:00:19.000Z | 32 | |
| amp_e56929 | Rk-Al4gjFrnePO5Vi2d1kq.NjUxMjc4MzIINGlwYzdKMTU... | .completecoding.in | / | 2025-11-15T03:08:42.000Z | 93 | |
| amp_e56929_completecoding.in | Rk-Al4gjFrnePO5Vi2d1kq.NjUxMjc4MzIINGlwYzdKMTU... | .completecoding.in | / | 2025-11-13T15:33:29.000Z | 110 | |
| SESSIONID | C2A93FEDBFC03164BE4A45FA51E74F2 | learn.completecoding.in | / | Session | 41 | ✓ |
| c_login_token | 1695797312295 | learn.completecoding.in | / | 2025-12-18T09:01:00.498Z | 26 | ✓ |
| id | 5a3e5c2c-939d-4b02-a1d9-a8a793efcf50 | learn.completecoding.in | / | 2025-12-18T09:00:18.028Z | 38 | ✓ |
| org.springframework.web.servlet.i18n.CookieLocaleResolver.LOC... | en | learn.completecoding.in | / | Session | 66 | |



18.2 Adding Login Functionality

1. Add a login button to the nav bar pointing to /login
2. Create a auth router to handle login related routes, register the new router in app.js
3. Create a auth controller to handle GET request for /login and return a UI that does the following:
 - a. Accepts email and password
 - b. Has a Login button that submits the form to /login with a POST request.
4. Add a post login path in router and handler in controller.
5. Assume the person logged in and redirect them to the home page.



18.2 Adding Login Functionality

1.

```
<div>
  <a href="/login" class="bg-blue-600 hover:bg-blue-700 text-white font-semibold py-2.5
    px-6 rounded-lg transition duration-300 ease-in-out transform hover:scale-105 shadow-md">
    Login
  </a>
</div>
```



18.2 Adding Login Functionality

2.

```
const express = require("express");
const authRouter = express.Router();
const authController = require("../controllers/authController");

authRouter.get("/login", authController.getLogin);

exports.authRouter = authRouter;
```

```
app.use("/host", hostRouter);
app.use(authRouter);
```



18.2 Adding Login Functionality

3.

```
exports.getLogin = (req, res, next) => {
  ...
  res.render("auth/login", { pageTitle: "Login" });
};
```



18.2 Adding Login Functionality

3.

```
<%- include('../partials/head') %>
</head>
<body class="min-h-screen bg-gray-100">
  <%- include('../partials/nav') %>
  <h1 class="text-4xl font-bold text-blue-600 mb-8 text-center mt-8">Login Here</h1>
  <div class="flex justify-center">
    <form action="/auth/login" method="POST" class="w-full max-w-md">
      <input
        type="email"
        name="email"
        placeholder="Enter your email"
        class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
      />
      <input
        type="password"
        name="password"
        placeholder="Enter your password"
        class="w-full px-4 py-2 mb-4 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
      />
      <div class="flex justify-center">
        <input
          type="submit"
          value="Login"
          class="bg-blue-500 hover:bg-blue-600 text-white font-semibold py-2 px-4 rounded-lg transition duration-300 ease-in-out transform hover:scale-105 cursor-pointer"
        >
      </div>
    </form>
  </div>
</main>
</body>
</html>
```



18.2 Adding Login Functionality

4, 5.

```
authRouter.get("/login", authController.getLogin);
authRouter.post("/login", authController.postLogin);
```

```
exports.postLogin = (req, res, next) => {
  console.log(req.body);
  res.redirect("/");
};
```



18.3 Checking Login State

1. Add a `isLoggedIn` field in the `req` object and use it everywhere else.
2. **Add** a condition in the navigation `ejs` file that no path other than home and login should be visible until a user has logged in.
3. Fix all the render calls to send the flag
4. Also add a middleware for host routes that if the user is not logged in they should be redirected to the login page.



18.3 Checking Login State

1.

```
exports.postLogin = (req, res, next) => {
  console.log(req.body);
  req.isLoggedIn = true;
  res.redirect("/");
};
```



18.3 Checking Login State

2.

```
<% if (isLoggedIn) { %>
  <a href="/favourites" class="bg-blue-600 hover:bg-blue-700 text-white">
    Favourites
  </a>
  <a href="/host/host-homes" class="bg-blue-600 hover:bg-blue-700 text-white">
    Host Homes
  </a>
  <a href="/host/add-home" class="bg-blue-600 hover:bg-blue-700 text-white">
    Add Home
  </a>
<% } %>
```



18.3 Checking Login State

3.

```
res.render("host/edit-home", {  
  home: home,  
  editing: editing,  
  pageTitle: "Edit Your Home",  
  isLoggedIn: req.isLoggedIn,  
});
```



18.3 Checking Login State

4.

```
app.use(storeRouter);
app.use("/host", (req, res, next) => {
  if (!req.isLoggedIn) {
    return res.redirect("/login");
  }
  next();
});
app.use("/host", hostRouter);
```



18.4 Using a Cookie

1. Understand why a global variable would not work.
2. Set a cookie on successful login. See it in storage, also on the next request.
3. Read the cookie using syntax and Define a middleware to set this value to the request object.

```
console.log(req.get('Cookie').split('=')[1]);
```

4. Change the Cookie from the browser and see the result.



18.4 Using a Cookie

2.

```
exports.postLogin = (req, res, next) => {
  console.log(req.body);
  res.cookie("isLoggedIn", true);
  res.redirect("/");
};
```

3.

```
app.use((req, res, next) => {
  req.isLoggedIn = req.get('Cookie')?.split('=')[1] || false;
  console.log(req.isLoggedIn);
  next();
});
```



18.4 Using a Cookie

4.

The screenshot shows the Chrome DevTools interface with the Application tab selected. On the left, the Storage section is expanded, showing Local storage, Session storage, IndexedDB, and Cookies. The Cookies section is also expanded, showing a cookie for the URL `http://localhost:3001`. The main panel displays a table with two columns: Name and Value. A single row is selected, showing the name `isLoggedIn` and the value `false`.

| Name | Value |
|------------|-------|
| isLoggedIn | false |

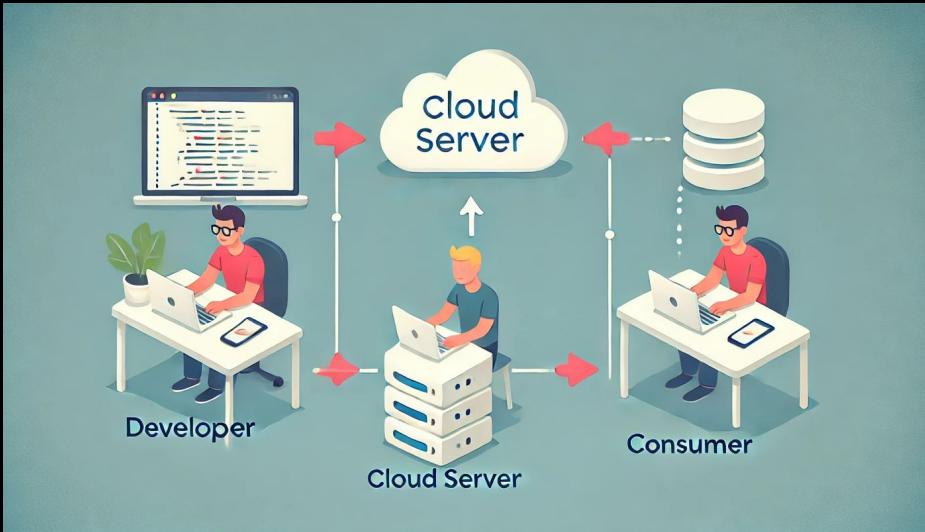


22. Deploying Apps

1. What is Deployment?
2. Cloud vs Local Deployments
3. Using Environment Variables
4. Secure Response Headers
5. Use Compressed Assets
6. Request-Response Logging
7. Using SSL/TLS Encryption
8. Using Hosting Provider



22.1 What is Deployment?



- Releasing software to users.
- Making software operational.
- Transitioning from testing to live environments.
- Using automated or manual processes.
- Delivering new features and improvements.



22.2 Cloud vs Local Deployments

| Aspect | Cloud Deployment | Local Deployment (On-Premises) |
|----------------|---|---|
| Infrastructure | Hosted on third-party cloud providers' servers | Hosted on organization's own servers and hardware |
| Scalability | Easily scalable; resources adjusted quickly based on demand | Limited scalability; requires physical hardware upgrades |
| Cost | Pay-as-you-go with lower upfront costs (Operational Expenses) | High upfront investment in hardware (Capital Expenses) |
| Maintenance | Managed by cloud provider; less effort for the organization | Managed internally; organization handles all maintenance |
| Accessibility | Accessible anywhere with internet connectivity | Typically accessible only within the organization's network |
| Control | Less control over underlying infrastructure | Full control over hardware and software configurations |



22.3 Using Environment Variables

```
const MONGO_DB_URL =
`mongodb+srv://${process.env.MONGO_DB_USERNAME}:${process.env.MONGO_DB_PASSWORD}@kgcluster.
ie6mb.mongodb.net/${process.env.MONGO_DB_DATABASE}`;

const PORT = process.env.PORT || 3000;
mongoose.connect(MONGO_DB_URL).then(() => {

const SEND_GRID_KEY = process.env.SEND_GRID_KEY;

const forgotEmail = {
  to: email,
  from: process.env.SENDER_EMAIL || 'contact@completecoding.in',
  subject: 'Here is your OTP to reset your Password',
```



22.3 Using Environment Variables

```
# Server
PORT=3000

# MongoDB
MONGO_DB_USERNAME=root
MONGO_DB_PASSWORD=root
MONGO_DB_DATABASE=airbnb

# Email
SEND_GRID_KEY=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
SENDER_EMAIL=contact@completecoding.in
```



```
1 .env
2
3 .DS_Store
```



22.3 Using Environment Variables

.env.example U X

5 NodeJs and ExpressJs > 18 deployment > .env.example

```
1 # Server Configuration
2 # Port number for the server to listen on (e.g. 3000)
3 PORT=
4
5 # MongoDB Database Configuration
6 # Username for MongoDB authentication
7 MONGO_DB_USERNAME=
8 # Password for MongoDB authentication
9 MONGO_DB_PASSWORD=
10 # Name of the MongoDB database
11 MONGO_DB_DATABASE=
12
13 # Email Service Configuration
14 # SendGrid API key for sending emails
15 # Get this from your SendGrid account dashboard
16 SEND_GRID_KEY=
17 # Email address used as the sender for all outgoing emails
18 # Format: example@domain.com
19 SENDER_EMAIL=
```



22.3 Using Environment Variables

npm Search packages Sign Up Sign In

dotenv TS

16.4.5 · Public · Published 9 months ago

[Readme](#) [Code](#) Beta [0 Dependencies](#) [50,765 Dependents](#) [86 Versions](#)

Announcing dotenvx. run anywhere, multi-environment, encrypted envs.

Dotenv is supported by the community.

Special thanks to:

warp

Warp is a blazingly fast, Rust-based terminal reimagined to work like a modern app. Get more done in the CLI with real text editing, block-based output, and AI command search.

WorkOS

Your App, Enterprise Ready.

Add Single Sign-On, Multi-Factor Auth, and more, in minutes instead of months.

Install

`> npm i dotenv`

Repository

[github.com/motdotla/dotenv](#)

Homepage

[github.com/motdotla/dotenv#...](#)

Fund this package

Weekly Downloads

44,221,1

49

Version License

16.4.5 BSD-2-Clause

```
prashantjain@Prashants-MacBook-Pro 18 deployment % npm install dotenv  
added 1 package, and audited 299 packages in 581ms  
51 packages are looking for funding  
  run `npm fund` for details  
1 high severity vulnerability  
To address all issues, run:  
  npm audit fix  
Run `npm audit` for details.
```

5 NodeJs and ExpressJs > 18 deployment > `js` app.js > ...

```
1   require("dotenv").config();  
2
```



22.3 Using Environment Variables

```
!+ .env.development  
!+ .env.example  
!+ .env.production
```

U U U

```
const environment = process.env.NODE_ENV || 'development';
require("dotenv").config({
  path: `./.env.${environment}`
});
```

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "tailwind": "npx tailwindcss -i ./public/styles/input.css -o ./public/styles/output.css
  --watch",
  "dev": "concurrently \"npm run tailwind\" \"NODE_ENV=development nodemon app.js\"",
  "start": "NODE_ENV=production node app.js"
},
```

For Windows

```
"scripts": {
  // ... existing code ...
  "tailwind": "npx tailwindcss -i ./public/styles/input.css -o ./public/styles/output.css --watch",
  "dev": "concurrently \"npm run tailwind\" \"set NODE_ENV=development&& nodemon app.js\"",
  "start": "set NODE_ENV=production&& node app.js"
}
```



22.4 Secure Response Headers

npm Search packages Sign Up Sign In

helmet ts

8.0.0 • Public • Published 2 months ago

[Readme](#) [Code](#) Beta [0 Dependencies](#) [5,441 Dependents](#) [132 Versions](#)

Helmet

Help secure Express apps by setting HTTP response headers.

```
import helmet from "helmet";  
  
const app = express();  
  
app.use(helmet());
```

Helmet sets the following headers by default:

- [Content-Security-Policy](#) : A powerful allow-list of what can happen on your

prashantjain@Prashants-MacBook-Pro 18 deployment % npm install helmet
added 1 package, and audited 300 packages in 615ms
51 packages are looking for funding
run `npm fund` for details
1 **high** severity vulnerability
To address all issues, run:
npm audit fix

Install `> npm i helmet`

Repository github.com/helmetjs/helmet

Homepage helmetjs.github.io/

Weekly Downloads 3,313,977



22.4 Secure Response Headers

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

const app = express();
app.use(helmet());
```

| ▼Response Headers | |
|------------------------------------|---|
| | Raw |
| Connection: | keep-alive |
| Content-Security-Policy: | default-src 'self';base-uri 'self';font-src 'self' https: data:;form-action 'self';frame-ancestors 'self';img-src 'self' data:;object-src 'none';script-src 'self';script-src-attr 'none';style-src 'self' https: 'unsafe-inline';upgrade-insecure-requests |
| Cross-Origin-Opener-Policy: | same-origin |
| Cross-Origin-Resource-Policy: | same-origin |
| Date: | Thu, 28 Nov 2024 11:30:24 GMT |
| Etag: | W/"2a7a-RJYQ4tZOJFrdD8vfcF7xdRYNqB8" |
| Keep-Alive: | timeout=5 |
| Origin-Agent-Cluster: | ?1 |
| Referrer-Policy: | no-referrer |
| Strict-Transport-Security: | max-age=31536000; includeSubDomains |
| X-Content-Type-Options: | nosniff |
| X-Dns-Prefetch-Control: | off |
| X-Download-Options: | noopen |
| X-Frame-Options: | SAMEORIGIN |
| X-Permitted-Cross-Domain-Policies: | none |
| X-Xss-Protection: | 0 🖊 |



22.5 Use Compressed Assets

Screenshot of the npm package page for `compression`.

The page shows the following details:

- Search bar:** Search packages
- Buttons:** Search, Sign Up, Sign In
- Package Information:** `compression`, version 1.7.5, Public, Published a month ago.
- Links:** Readme, Code (Beta), 7 Dependencies, 7,925 Dependents, 37 Versions
- Description:** Node.js compression middleware.
- Supported Codings:** deflate, gzip
- Install:** npm i compression
- Repository:** github.com/expressjs/compression
- Homepage:** github.com/expressjs/compression
- Downloads:** 20,090,348 weekly downloads
- Version:** Version
- License:** License

```
prashantjain@Prashants-MacBook-Pro ~ % npm install compression
added 3 packages, and audited 303 packages in 654ms
51 packages are looking for funding
  run `npm fund` for details
1 high severity vulnerability
To address all issues, run:
  npm audit fix
Run `npm audit` for details.
```

```
const compression = require("compression");
const app = express();
app.use(helmet());
app.use(compression());
```



22.5 Use Compressed Assets

```
//app.use(compression());
```

```
app.use(compression());
```

| Name | Status | Type | Ini... | Size |
|----------------|--------|------------|----------|---------|
| localhost | 200 | document | Ot... | 11.4 kB |
| output.css | 200 | stylesheet | (inde... | 22.2 kB |
| house1.png | 200 | png | (inde... | 388 kB |
| house7.png | 200 | png | (inde... | 331 kB |
| house2.png | 200 | png | (inde... | 350 kB |
| 2024-11-26T... | 200 | jpeg | (inde... | 1.3 MB |
| favicon.ico | 404 | text/html | Ot... | 2.1 kB |

| Name | Status | Type | Ini... | Size |
|----------------|--------|------------|----------|--------|
| localhost | 200 | document | Ot... | 2.3 kB |
| house2.png | 200 | png | (inde... | 350 kB |
| 2024-11-26T... | 200 | jpeg | (inde... | 1.3 MB |
| output.css | 200 | stylesheet | (inde... | 6.0 kB |
| house1.png | 200 | png | (inde... | 388 kB |
| house7.png | 200 | png | (inde... | 331 kB |



22.6 Request-Response Logging

npm Search packages Sign Up Sign In

morgan DT

1.10.0 • Public • Published 5 years ago

[Readme](#) [Code](#) Beta [5 Dependencies](#) [9,399 Dependents](#) [26 Versions](#)

morgan

npm v1.10.0 downloads 22.3M/month travis 404 coverage 100%

HTTP request logger middleware for node.js

Named after **Dexter**, a show you should not watch until completion.

API

```
var morgan = require('morgan')
```

prashantjain@Prashants-MacBook-Pro 18 deployment % npm install morgan
added 4 packages, and audited 307 packages in 808ms
51 packages are looking for funding
run `npm fund` for details
1 **high** severity vulnerability
To address all issues, run:
npm audit fix
Run `npm audit` for details.



22.6 Request-Response Logging

```
const fs = require("fs");
const morgan = require("morgan");

const accessLogStream = fs.createWriteStream(path.join(__dirname, "access.log"), {
  flags: "a",
});
app.use(morgan("combined", { stream: accessLogStream }));
```

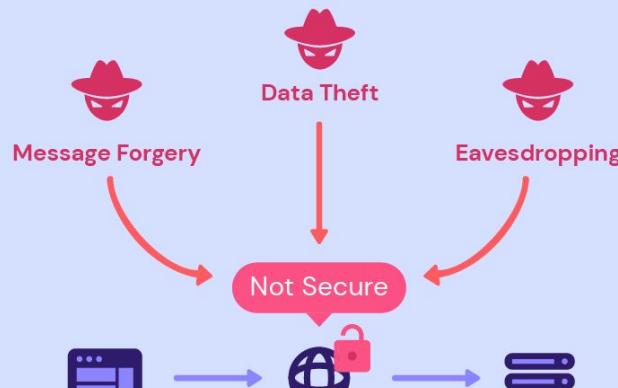
5 NodeJs and ExpressJs > 18 deployment > access.log

```
1 ::1 - - [28/Nov/2024:11:43:55 +0000] "GET / HTTP/1.1" 304 - "http://localhost:3000/homes/6739886801b45364227c9428" "Mozilla/5.0
(Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"
2 ::1 - - [28/Nov/2024:11:43:55 +0000] "GET /styles/output.css HTTP/1.1" 304 - "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"
3 ::1 - - [28/Nov/2024:11:43:55 +0000] "GET /images/house1.png HTTP/1.1" 304 - "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"
4 ::1 - - [28/Nov/2024:11:43:55 +0000] "GET /images/house7.png HTTP/1.1" 304 - "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"
5 ::1 - - [28/Nov/2024:11:43:55 +0000] "GET /images/house2.png HTTP/1.1" 304 - "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36"
6 ::1 - - [28/Nov/2024:11:43:55 +0000] "GET /uploads/2024-11-26T15:17:10.226Z-1449px-Mona_Lisa,_by_Leonardo_da_Vinci,
_from_C2RMF_retouch.jpg HTTP/1.1" 304 - "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.0.0 Safari/537.36"
7 ::1 - - [28/Nov/2024:11:43:55 +0000] "GET /uploads/2024-11-26T15:17:10.226Z-1449px-Mona_Lisa,_by_Leonardo_da_Vinci,
_from_C2RMF_retouch.jpg HTTP/1.1" 200 1286409 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.0.0 Safari/537.36"
```



22.7 Using SSL/TLS Encryption

HTTP No Encryption (No SSL)



HTTPS Secured with SSL

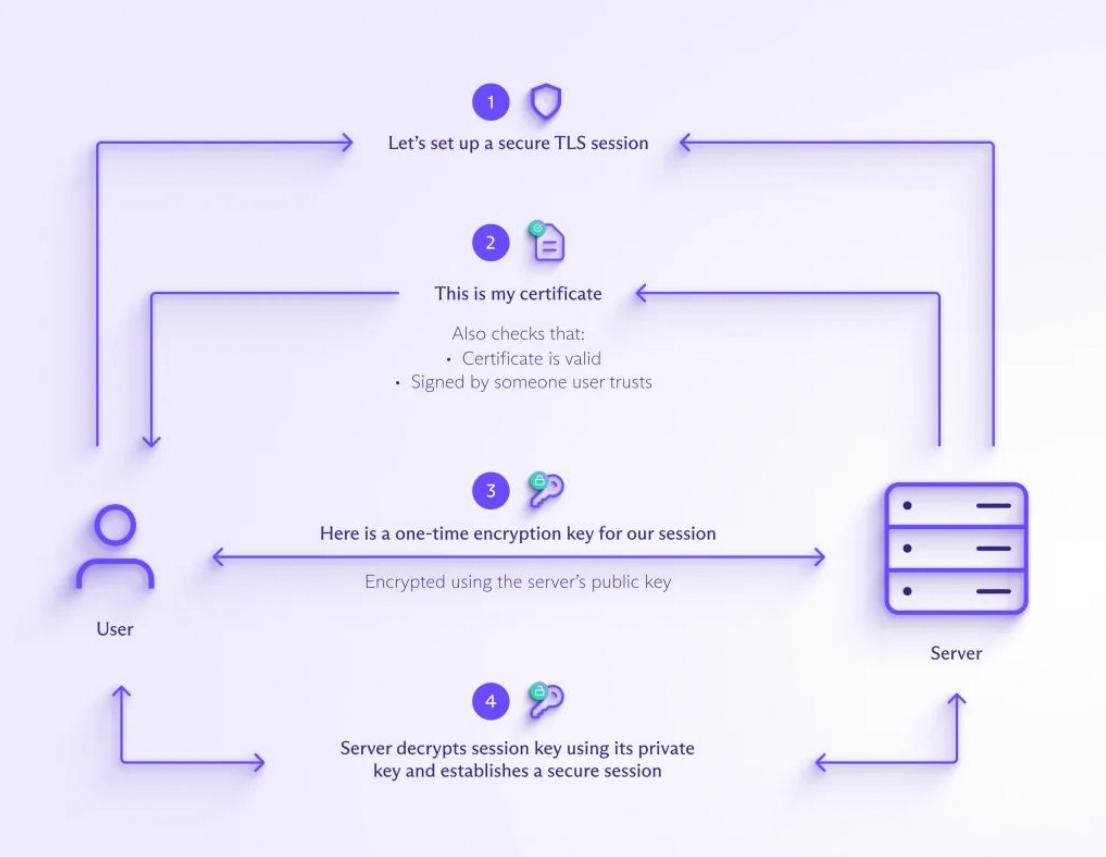


Data, such as a user password and a user ID, is **visible** to anyone.

Data, such as a user password and a user ID, is **encrypted** to anyone



22.7 Using SSL/TLS Encryption





22.8 Using Hosting Provider

Microsoft | Azure

Get started with Azure

Build in the cloud with an Azure account

Get started creating, deploying, and managing applications—across multiple cloud, on-premises, and at the edge—with scalable and cost-efficient Azure services.

[Try Azure for free](#) [Pay as you go](#)

The screenshot shows the Microsoft Azure landing page. On the left, there's a large call-to-action section with the heading 'Build in the cloud with an Azure account'. Below it, a paragraph describes getting started with creating, deploying, and managing applications across various environments using Azure services. At the bottom of this section are two buttons: 'Try Azure for free' (in a green box) and 'Pay as you go' (in a white box). To the right of this main section is a large image of a computer monitor displaying the Azure portal interface, which includes a 'Start with an Azure free trial' card and a 'Manage Microsoft Entra ID' card. The overall background is light gray, and the Azure branding is prominent.

Choose the Azure account that's right for you

Pay as you go or try Azure free for up to 30 days. There's no upfront commitment—cancel anytime.



22.8 Using Hosting Provider

[Microsoft](#) | [Azure](#) [Explore](#) [Products](#) [Solutions](#) [Pricing](#) [Partners](#) [Resources](#) [Search](#) [Learn](#) [Support](#) [Contact Sales](#) [Get started with Azure](#) [Sign in](#)

You're ready to start with Azure

[Go to Azure portal](#)

Learn how to get started quickly, with the help of an Azure expert

[Join a Q&A session](#) with an Azure technical expert in a live webinar format. Participate in discussions with peers and get your questions answered. Schedule a session anytime you have questions or are looking for inspiration.

Watch the [on-demand demo](#) series to learn how to:

- Navigate & work in the Azure portal.
- Architect your solutions & organize resources.
- Estimate & manage costs.
- Deploy common services, including virtual machines, web apps, and SQL databases.

The demo series is available in English, Chinese (Traditional), German, Korean, Portuguese (Brazilian), Spanish, and Turkish. Captions are available in many other languages.