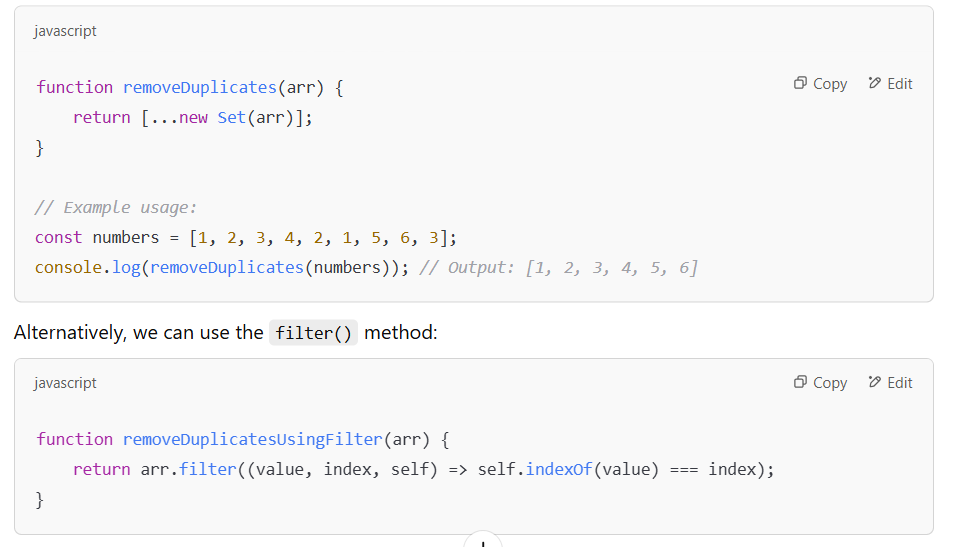
**1) Write JS code to remove duplicate values from an array?**

How would you remove duplicate values from an array in JavaScript?

There are multiple ways to remove duplicates from an array in JavaScript.

One of the most efficient ways is using the Set object since it only stores unique values.

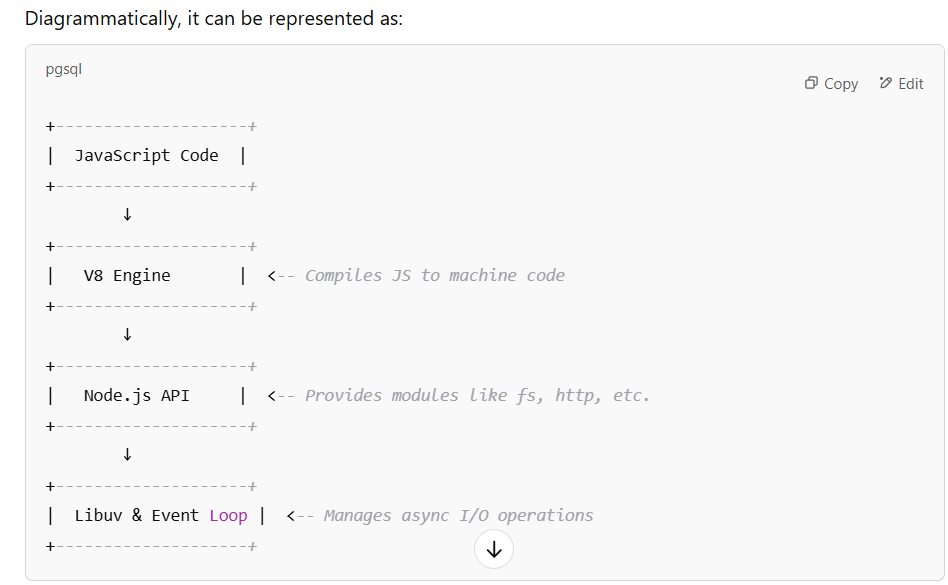


**2) Define the underlying Architecture of Node?**

Yes, Node.js is built on the **V8 JavaScript engine** and follows an **event-driven, non-blocking I/O model**.

The key components of Node.js architecture are:

1. **V8 Engine** – It compiles JavaScript code into machine code.
2. **Libuv** – It provides the event loop and handles asynchronous operations.
3. **Event Loop** – The core of Node.js that enables non-blocking I/O.
4. **C++ Bindings** – Some low-level operations are handled in C++ for performance.
5. **Thread Pool** – Node.js uses worker threads for CPU-intensive tasks.



**3) What is middleware? Write a function to explain middleware?**

Middleware is a function in Express.js that executes **before the final request handler**.

It has access to the **request (req), response (res), and next()** function, which passes control to the next middleware.



**4) What is the Event Loop?**

The **Event Loop** is the heart of Node.js. It allows **non-blocking asynchronous operations** by managing I/O tasks efficiently.

JS is **single-threaded**, but Node.js can handle multiple asynchronous operations using the Event Loop.

The Event Loop has different **phases**:

**1. Timers Phase** – Executes setTimeout() and setInterval().

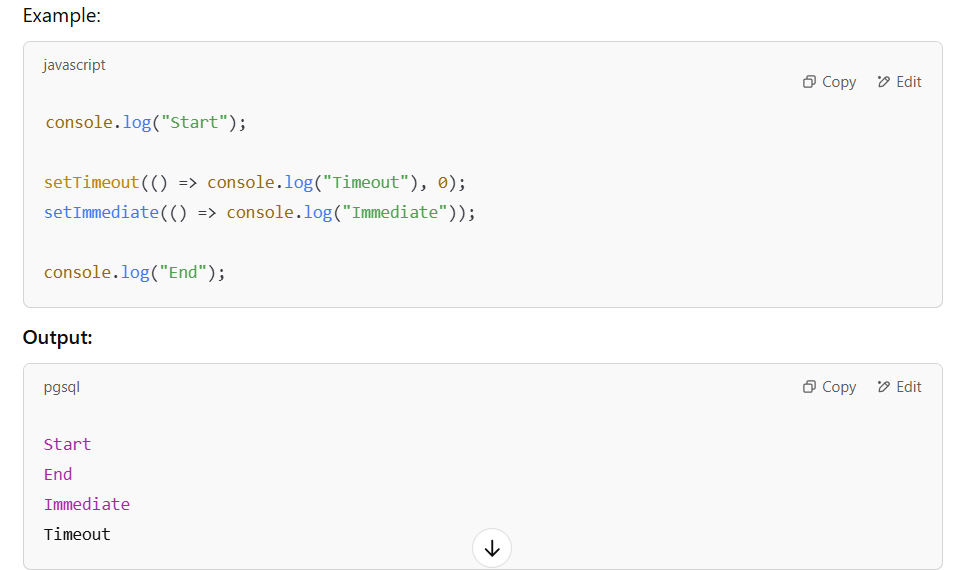
2.**Pending Callbacks Phase** – Executes I/O callbacks.

3.**Idle, Prepare Phase** – Internal use.

4.**Poll Phase** – Fetches new I/O events and executes callbacks.

5.**Check Phase** – Executes setImmediate() callbacks.

6.**Close Callbacks Phase** – Handles closed events like socket.on('close', callback).

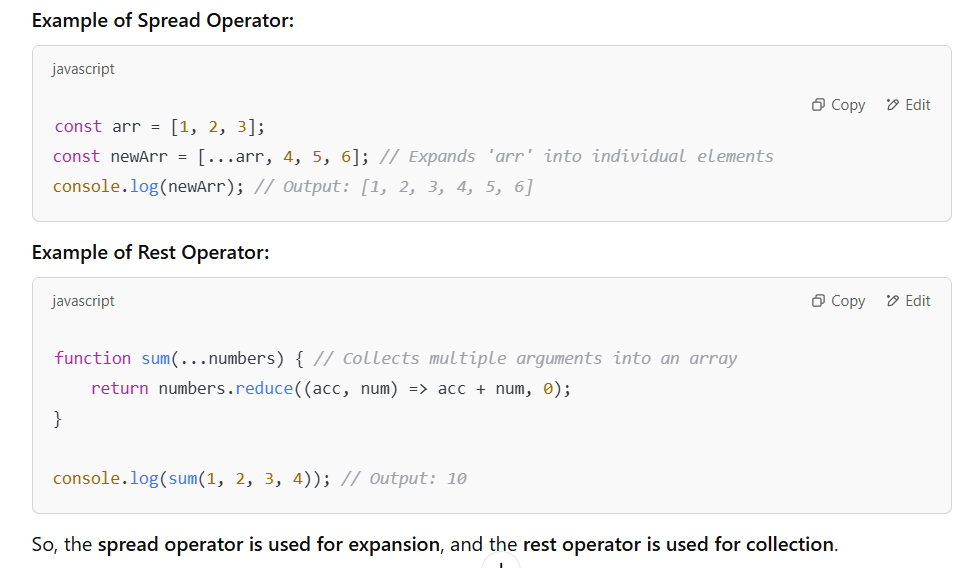


This shows how the Event Loop processes operations asynchronously.

**5) What is the spread and rest operator?**

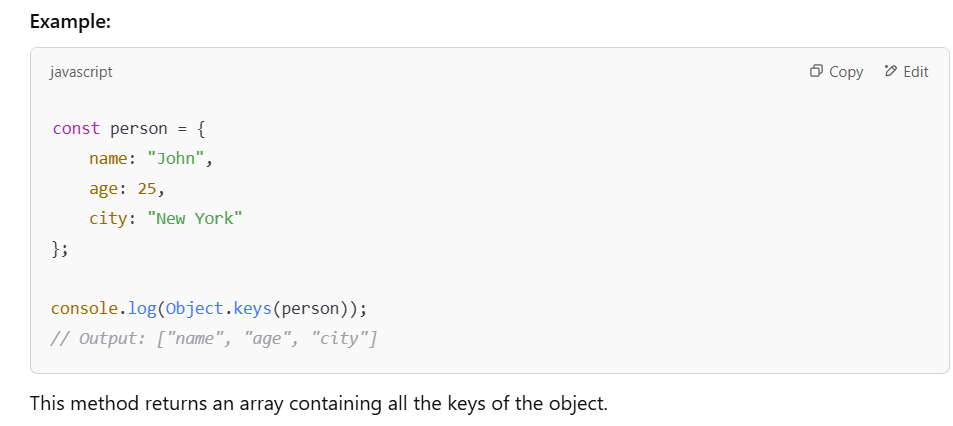
The **spread (...)** and **rest (...)** operators both use ..., but they serve different purposes:

* **Spread Operator (...)**: Expands an array or object into individual elements.
* **Rest Operator (...)**: Collects multiple values into an array.



**6) How to Fetch and Print Out All Keys from an Object?**

In JavaScript, we can use Object.keys() to fetch all the keys from an object.



**7) What is Node Package Manager (NPM)?**

NPM stands for **Node Package Manager**. It is the default package manager for **Node.js** and is used to:

1. **Install, update, and manage dependencies** for Node.js applications.
2. Provide access to **thousands of open-source libraries**.
3. Manage **global and local package installations**.



**8) What is the Importance of the package.json File?**

The package.json file is the **heart of any Node.js project**.

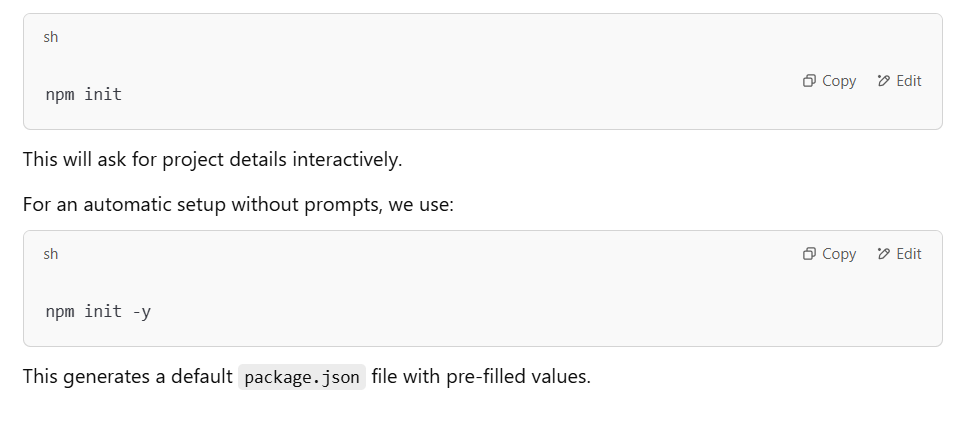
It contains metadata about the project, including:

1. **Project Information** – Name, version, description.
2. **Dependencies** – Lists the installed packages and their versions.
3. **Scripts** – Defines commands like npm start, npm test.
4. **Entry Point** – Specifies the main file to execute (e.g., index.js).



**9) How Can We Create a package.json File Through the Terminal?**

To create a package.json file, we use the command:



**10) Why Does Global Installation of Modules Require?**

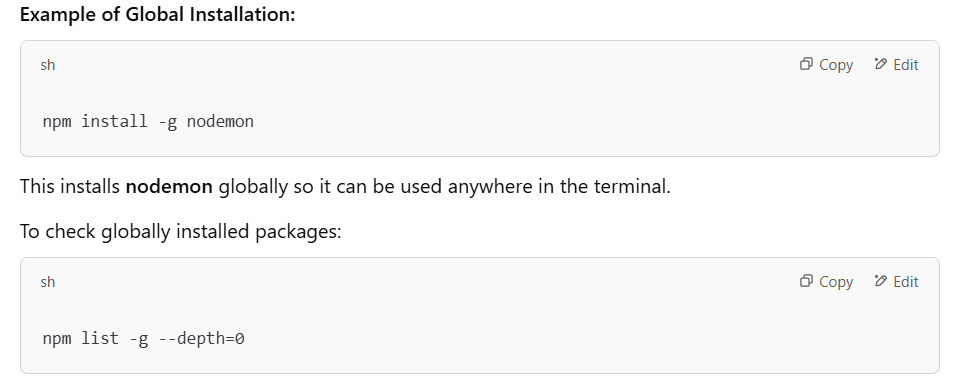
Global installation is required for modules that are **used system-wide** and not tied to a specific project.

**Key Reasons:**

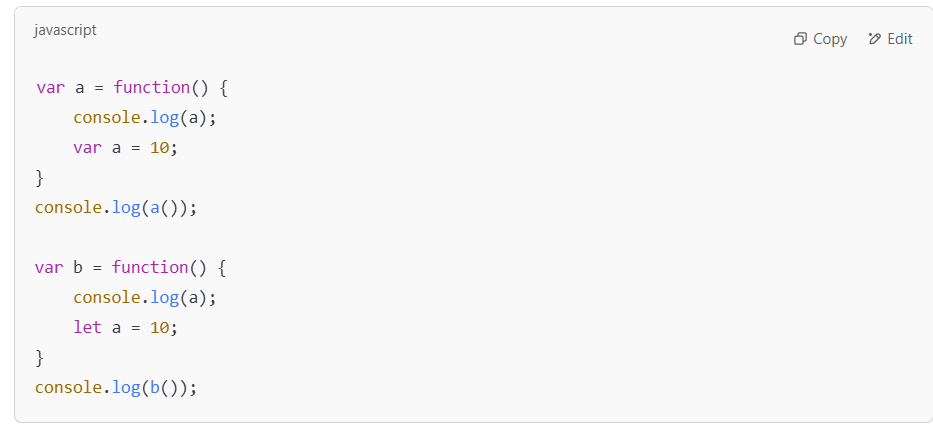
**1.Command-line utilities** – Some packages provide CLI tools.

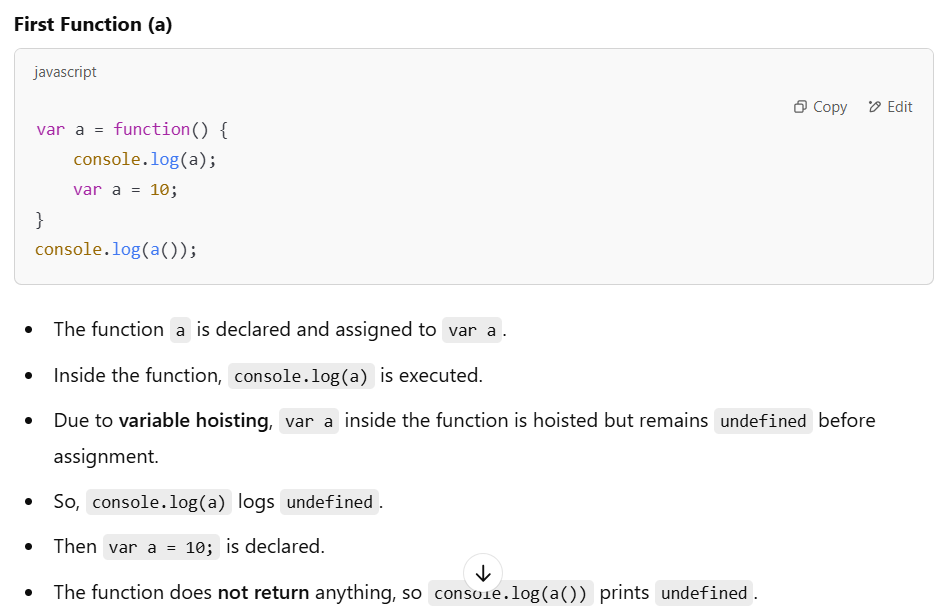
**2.Reusability** – Available for all projects on the system.

**3.Avoid redundant installations** – Saves space by avoiding duplicate local installations.

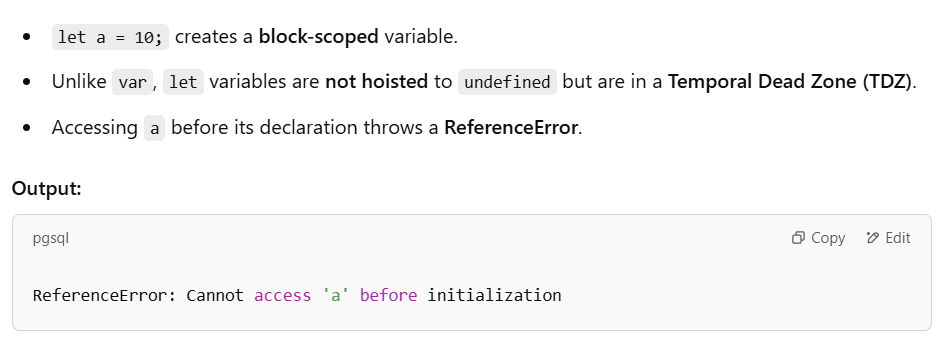


**11) What is the output of the below code?**

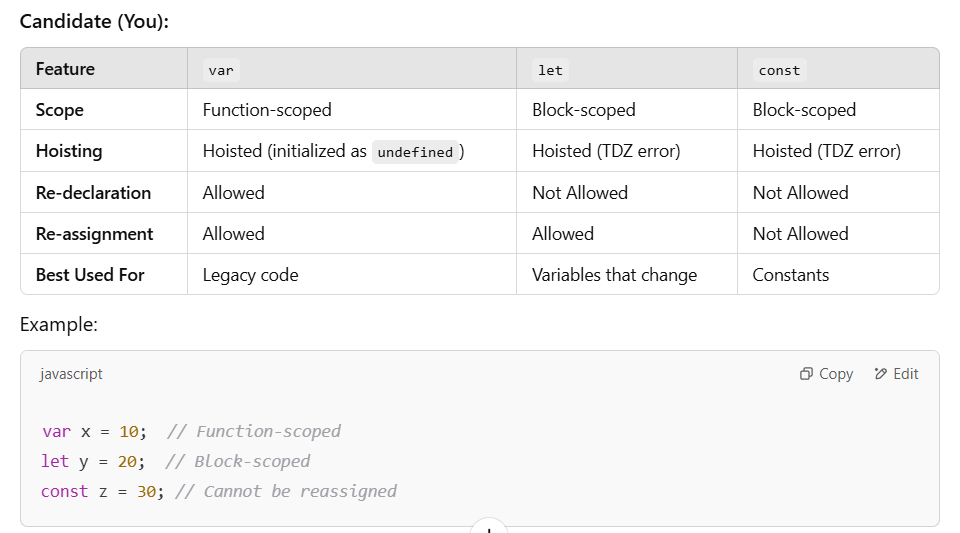








**12) What is the difference between var, let, and const?**



**13) What is a callback function and a higher-order function?**

A **callback function** is a function **passed as an argument** to another function and executed later.

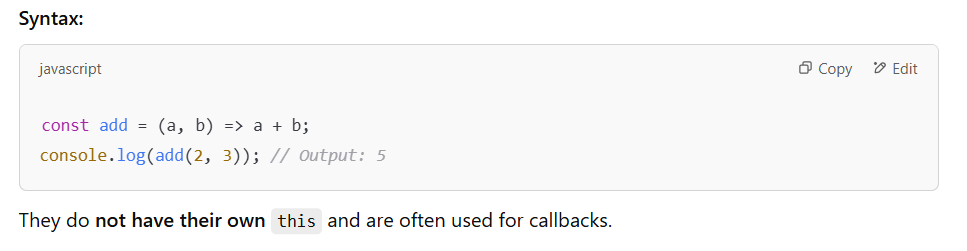


A **higher-order function** is a function that **takes another function as an argument or returns a function**.



**14) What is an arrow function? Can you write the syntax?**

Arrow functions (=>) are **shorter syntax** for defining functions in JavaScript.



**15) How to use variables a and b from one JS file in another?**

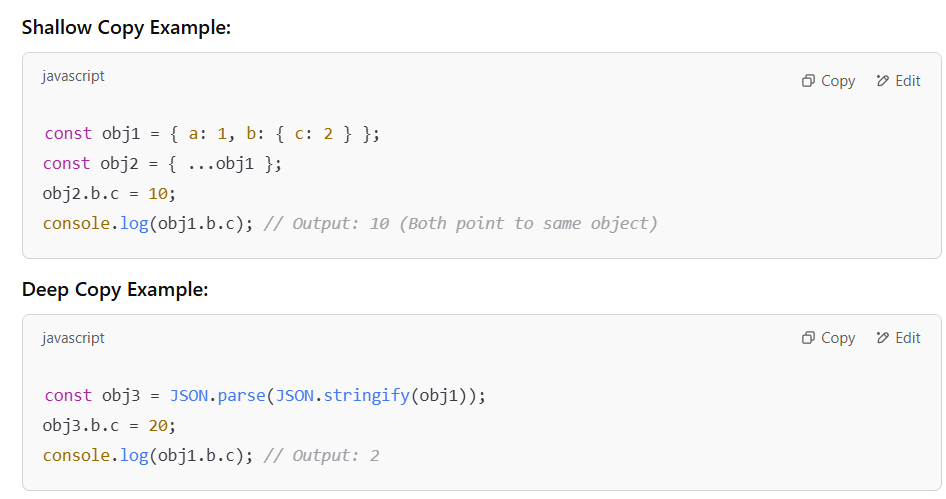
In **Node.js**, we use **module.exports** and **require**.



**16) What is deep copy and shallow copy?**

A **shallow copy** copies only the first level, while

a **deep copy** clones all nested objects.



**17) What is a prototype?**  
A **prototype** is an object from which other objects inherit properties.



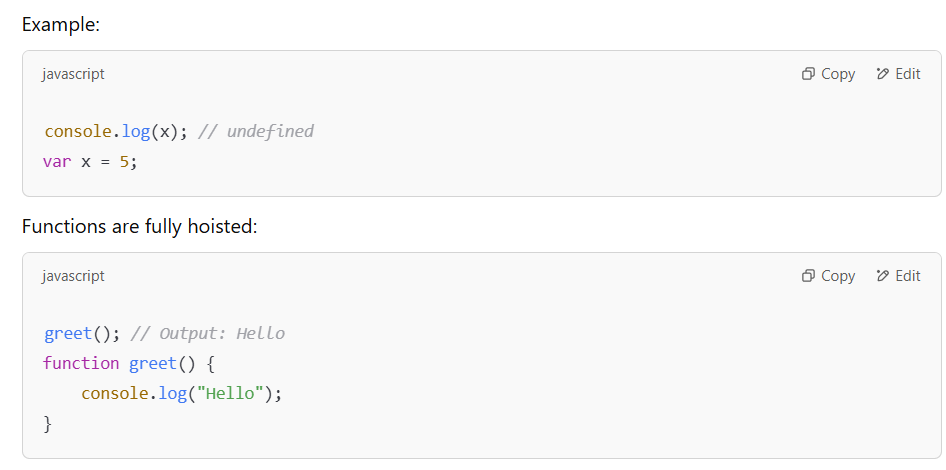
**18) What is a worker in Node.js?**

A **worker** in Node.js is a **separate thread** used to run CPU-intensive tasks **without blocking** the main thread.



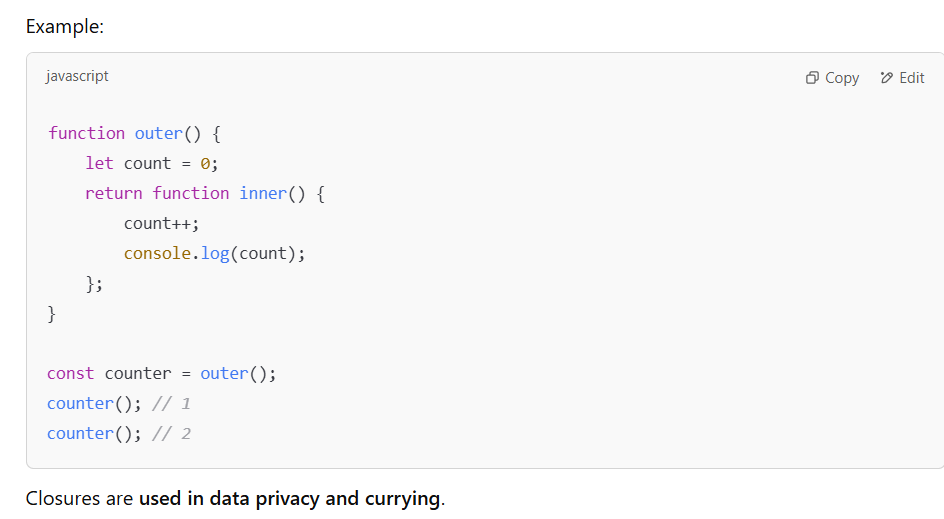
**19) What is hoisting?**

**Hoisting** moves function and variable declarations to the top of their scope **before execution**.



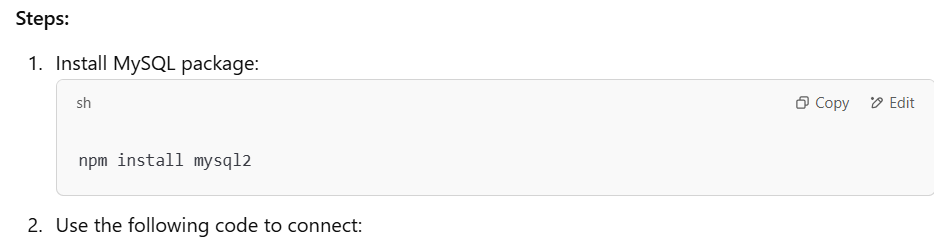
**20) What is a closure?**

A **closure** is a function that **remembers variables from its outer scope** even after execution.



**21) Write code to connect Node.js with MySQL database?**

We use the mysql2 package to establish a connection with MySQL in Node.js.





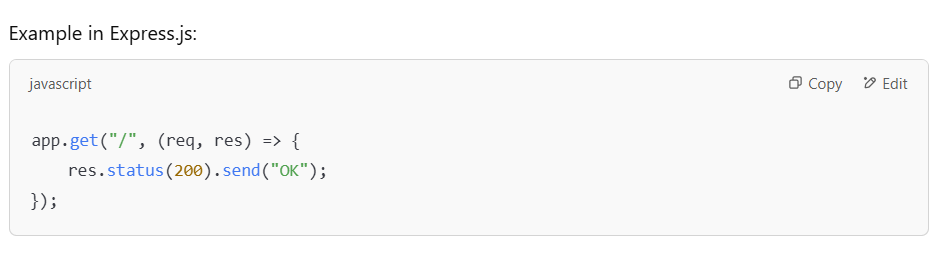


**22) What is an HTTP Status Code?**

HTTP status codes are **3-digit responses** sent by a server to indicate the outcome of an HTTP request.

**Common Status Codes:**

* **1xx (Informational):** 100 Continue
* **2xx (Success):** 200 OK, 201 Created
* **3xx (Redirection):** 301 Moved Permanently, 304 Not Modified
* **4xx (Client Errors):** 400 Bad Request, 401 Unauthorized, 404 Not Found
* **5xx (Server Errors):** 500 Internal Server Error, 503 Service Unavailable

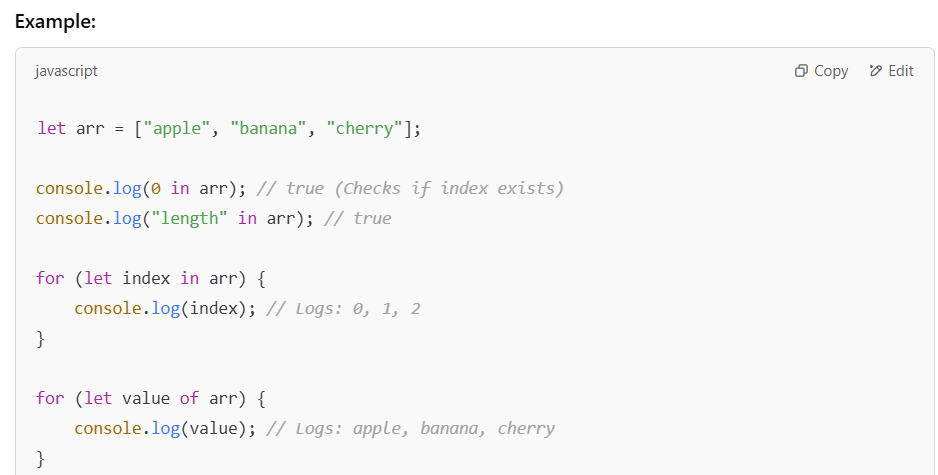


**23) Write code to count duplicate elements from an array**  
We can use an object to store the **frequency of elements**.



**24) What is the difference between in and of keywords in JavaScript?**

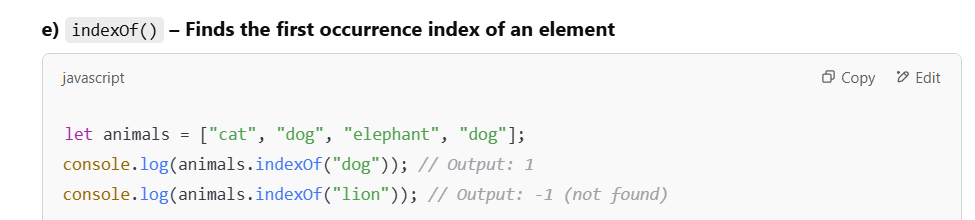
* **in** is used to **check for a property/index in an object or array**.
* **of** is used to **iterate over values of an array or iterable**.



**25) Explain Below Array Functions**







**26) How does the Event Loop work internally?**  
It is the mechanism that handles asynchronous operations in Node.js.

The Event Loop **repeats this sequence continuously** if tasks remain.

It follows a **fixed sequence of six phases**, continuously running:

**1.Timer Phase** – Executes setTimeout() and setInterval() callbacks.

**2.Pending Callbacks Phase** Executes deferred I/O callbacks (e.g., TCP errors).

**3.Idle, Prepare Phase** – Used internally by libuv.

**4.Poll Phase** – The core of the Event Loop, where:

* Node waits for new I/O events.
* If no timers are pending, it pauses here.
* Executes I/O callbacks if available.

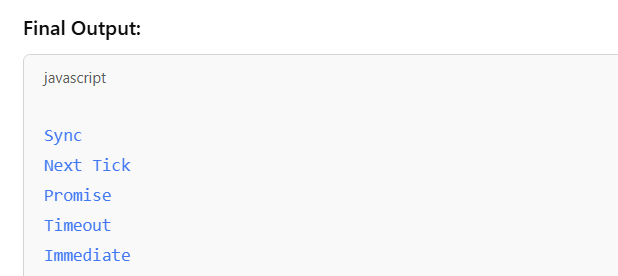
**5.Check Phase** – Executes setImmediate() callbacks.

**6.Close Callbacks Phase**: Executes process.exit() and stream/socket close events.

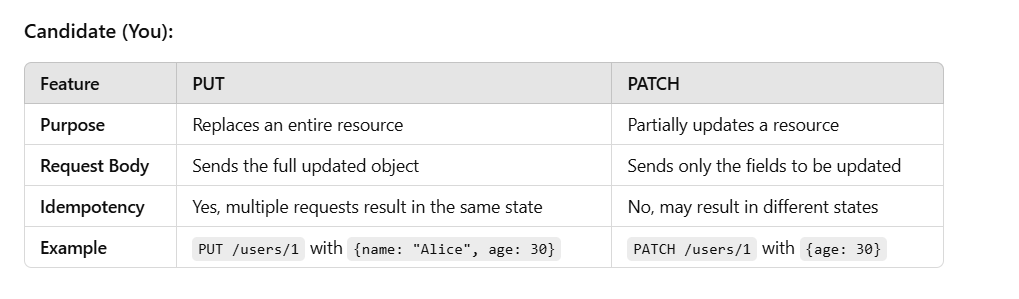
**27) What is the output of the following code?**

  
The execution order follows:

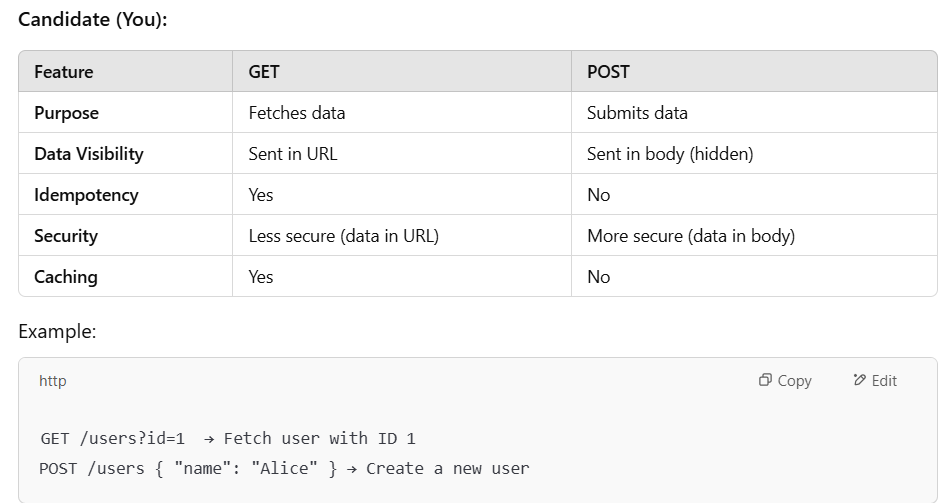
1. **Synchronous code executes first**, so Sync is printed.
2. **Microtasks (nextTick & Promise) run before the next phase of the Event Loop:**
   * process.nextTick() runs immediately after sync code (Next Tick).
   * Promise.resolve().then() runs next (Promise).
3. **Timers and setImmediate() run in the next event loop cycle:**
   * setTimeout() executes (Timeout).
   * setImmediate() executes (Immediate).



**28) Difference b/w PUT and PATCH HTTP methods?**



**29) Difference b/w POST and GET HTTP methods?**



**30) What is SQL Injection?**

SQL Injection is a **security vulnerability** where an attacker **injects malicious SQL queries** into an application.





**31) Why should we use ORM instead of writing raw SQL?**  
ORM (Object-Relational Mapping) **simplifies database interactions** by using JavaScript objects instead of raw SQL queries.

**Benefits:**

* **Avoids SQL injection** (safe queries).
* Easier database op’s (User.findAll() instead of SELECT \* FROM users).
* Database independence (works with MySQL, PostgreSQL, etc.).
* Reduces boilerplate code.



**32) How many types of modules are there in Node.js?**

**1.Core Modules (built-in)** – fs, http, path, etc.

**2.Local Modules (user-created)** – Custom modules using module.exports.

**3.Third-party Modules** – Installed via npm (e.g., express, mysql2).



**33) What is Debouncing?**

Debouncing **delays execution** of a function **until after a specified time** has passed since the last call.

**Use case:** Prevents excessive API calls on search input.



**34) How does throttling differ from debouncing?**  
Throttling **limits function execution** to **once in a set time interval**, even if triggered multiple times.



**35) What is MVC? Draw its architecture diagram?**  
MVC (Model-View-Controller) is a **design pattern** that separates an application into three components:

**1.Model** – Manages data and business logic.

**2.View** – Displays data to the user.

**3.Controller** – Handles user input and updates Model/View.



**36.What is the difference between package.json and package-lock.json.**

Both are essential files in a Node.js project, but they serve different purposes. Here's a breakdown of their differences:

**1. package.json**

**Purpose:** It contains metadata about the project, including dependencies, scripts, and other configurations.

**Role:** It defines the dependencies required for the project but does not lock their versions strictly.

**Versioning:** Uses version ranges (e.g., ^1.2.3, ~1.2.3) which allow updates within the specified range.

**Editable:** Manually editable by developers to add or update dependencies.

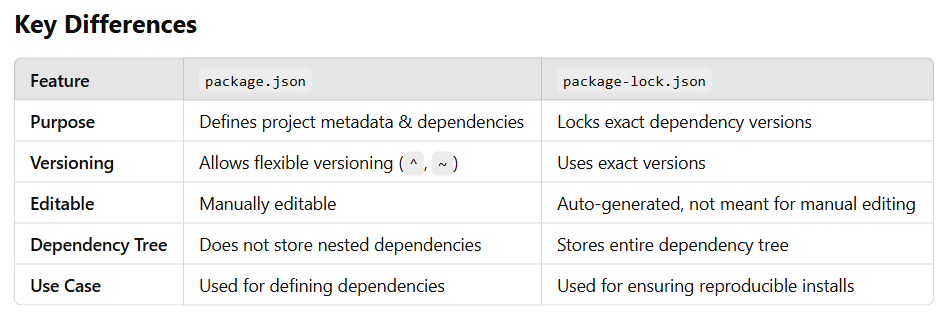
**2. package-lock.json**

**Purpose:** It locks the exact versions of dependencies (including nested dependencies) to ensure consistency across different environments.

**Role:** Automatically generated by npm to store the exact dependency tree at the time of installation.

**Versioning:** Uses exact versions (e.g., "express": "4.17.1") to avoid unintended updates.

**Not Manually Edited:** Should not be manually modified; it's managed by npm.



**When is package-lock.json important?**

Ensures **consistent dependency versions** across all environments.

Helps **faster installations** by avoiding version resolution on each install.

Reduces risks of **breaking changes** due to dependency updates.

**Should you commit package-lock.json?**

**Yes, for applications**: Ensures the same dependencies for all developers.

**No, for libraries/packages**: It can be ignored in published packages to allow flexibility for consumers.

37. Meaning of **app.use(express.urlencoded ({extended:false})).**

It is a middleware function in Express.js that is used to parse incoming requests with **URL-encoded payloads**.

**Breaking it Down**

**1.app.use()**:

This mounts middleware in Express. Middleware functions process incoming requests before they reach route handlers.

**2.express.urlencoded({...})**:

This is a built-in middleware in Express to parse application/x-www-form-urlencoded form data (used when submitting forms).

**3.{ extended: false }**:This option controls how the form data is parsed:

false:Uses the **querystring** library (supports only simple key-value pairs, no nested objects).

true:Uses the **qs** library (allows parsing of complex/nested objects).

**Use Case**

It is commonly used when handling form submissions where data is sent in URL-encoded format.



**When to Use extended: true?**



**38.Use case of "cors":, "dotenv":, "express": , "mysql2": ?**

These are commonly used Node.js packages in backend development.

**cors**

Enables Cross-Origin Resource Sharing (CORS) in Express applications.

**Need**:

By default, browsers enforce the **same-origin policy**, which restricts requests from different domains.

The cors package allows controlled access to resources from different origins.

**Use Case**: Required when building APIs that are consumed by frontend applications hosted on different domains.

**dotenv**

Loads environment variables from a .env file into process.env.

**Need**:

Keeps sensitive information (e.g., API keys, database credentials) outside of the source code.

**Use Case**:

Used for configuration management and security in Node.js applications.

**express**

A minimalist web framework for Node.js used to build APIs and web app’s.

**Need**:

Provides routing, middleware support, and an easy way to handle HTTP requests and responses.

**Use Case**: Used to build RESTful APIs and web servers.

**mysql2**

A MySQL client for Node.js with support for both **callbacks** and **Promises**.

**Need**:

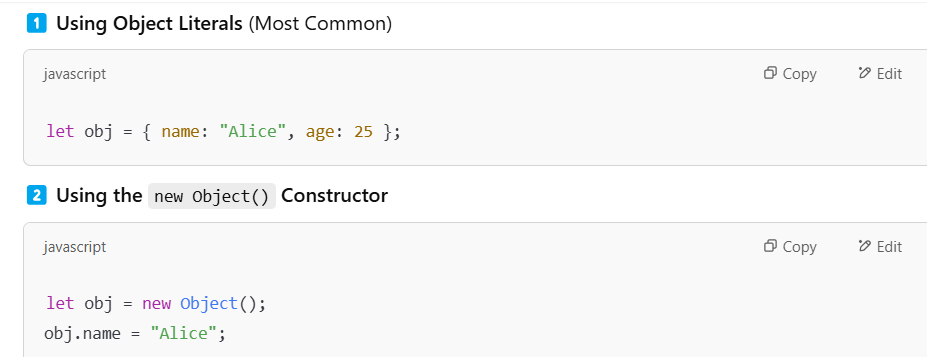
Allows interaction with MySQL databases to perform CRUD (Create, Read, Update, Delete) operations.

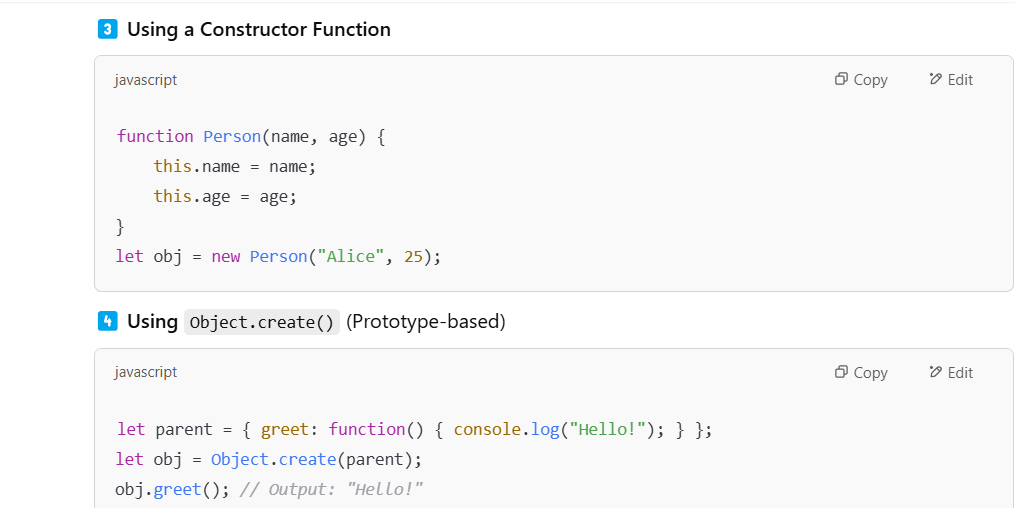
**Use Case**:

Used in applications that require relational database management with MySQL.

**39 What are the different ways to create an object in JS?**

There are **five** ways to create an object in JavaScript:







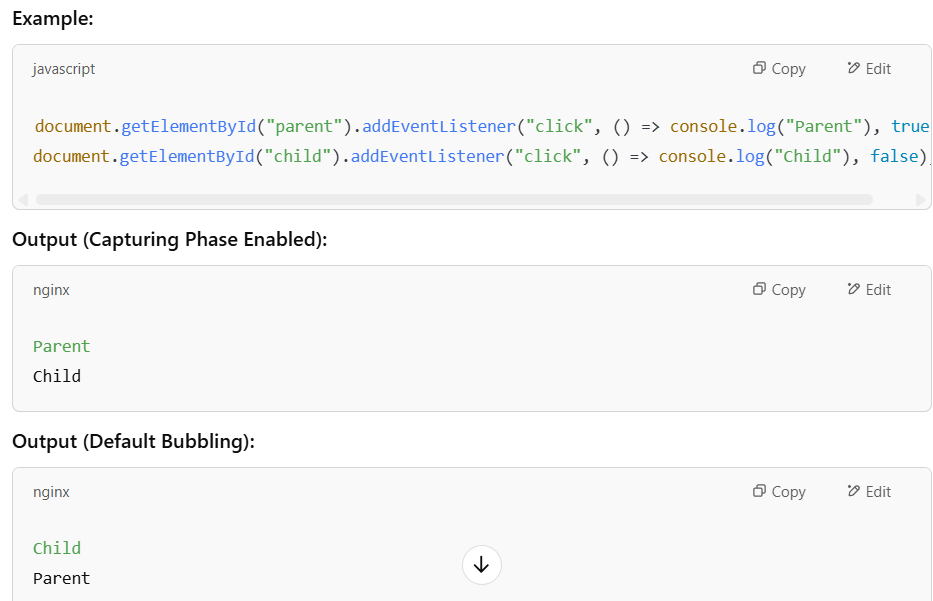
**40 Explain event bubbling and event capturing.**  
When an event happens on an element inside another element, it can be handled in **two phases**:

**Event Bubbling (Default)**:

* Event starts at the **target element** and moves **upwards** to the parent elements.
* Ex: Clicking a button inside a div triggers both the button’s and the div’s event handlers.

**Event Capturing (Trickle-down)**:

* Event starts at the **root element** and moves **downwards** to the target.
* It is enabled by passing true in addEventListener.



**41 What are Streams in Node.js?**

Streams in Node.js are **efficient ways to handle large amounts of data** without loading everything into memory.

There are **four types** of streams:

**Readable Streams** – Read data (fs.createReadStream()).

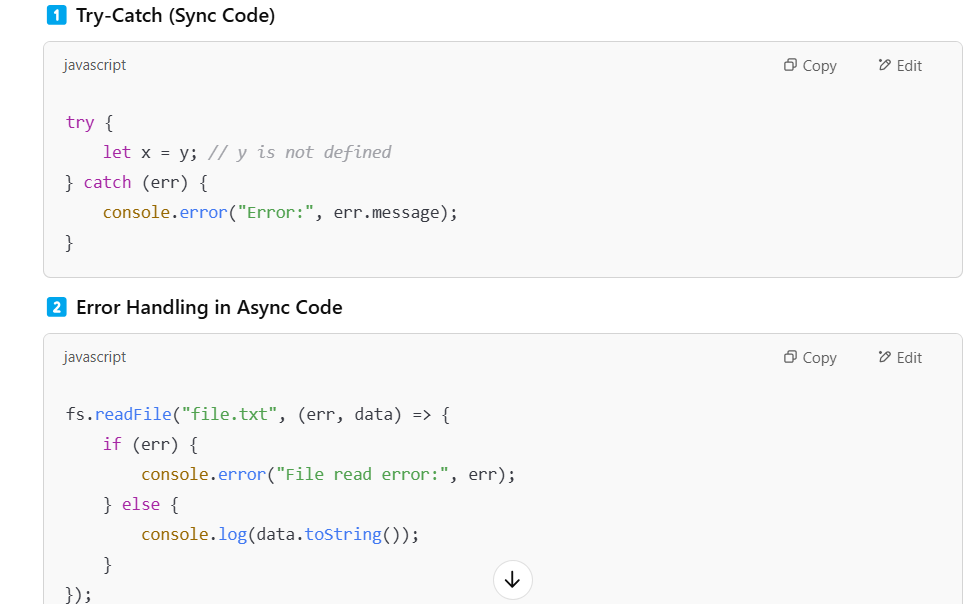
**Writable Streams** – Write data (fs.createWriteStream()).

**Duplex Streams** – Both read and write (net.Socket).

**Transform Streams** – Modify data (zlib.createGzip()).



**42 How would you handle error handling in Node.js applications?**

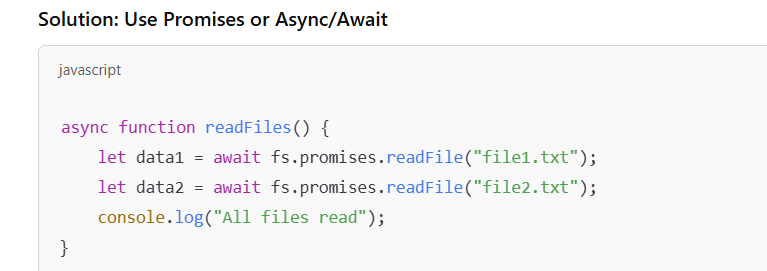




**43 What is Callback Hell and How to Avoid It?**

**Callback Hell** happens when we **nest multiple callbacks**, making the code hard to read and maintain.





**44 What is CORS and how do you handle it in a Node.js app?**

* **CORS (Cross-Origin Resource Sharing)** allows a browser to access resources from a different domain.

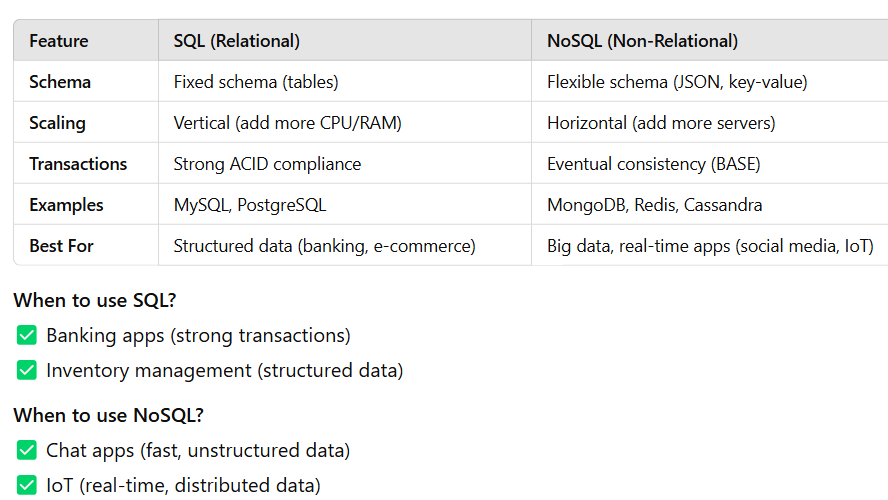
**Handling CORS in Express:**

const cors = require("cors"); app.use(cors()); // Allows all origins

**For Specific Domains:**

app.use(cors({ origin: "https://example.com" }));

**45. Difference Between SQL and NoSQL Databases? When to Use Each?**

****

**46. How do you read and write files asynchronously using the fs module in Node.js?**

* We use the fs.promises API to handle file operations asynchronously.



**47. What are the best practices for securing a Node.js app?**

* **Use Environment Variables**: Store secrets using dotenv.
* **Sanitize Inputs**: Prevent **SQL Injection/XSS** using libraries like express-validator.
* **Use Helmet.js**: Protect against security vulnerabilities.
* **Enable CORS Properly** – Restrict origins using cors.
* **Rate Limiting** – Prevent DDoS attacks with express-rate-limit.
* **Use HTTPS** – Encrypt data with SSL/TLS.
* **Session Management** – Secure cookies and sessions with express-session.
* **Validate JWT Tokens** – For authentication in APIs.

app.use(cors({ origin: "https://trusted.com" }));

const helmet = require("helmet"); app.use(helmet());

**48. Write a simple HTTP server using Node.js.**

* We use the built-in http module to create a server.

****

**49. What is app.use() in Express.js, and how is it used for middleware and static files?**

* **Middleware Usage**: app.use() applies middleware to all requests.

app.use((req, res, next) => {

console.log("Middleware executed");

next();

});

* **Serving Static Files**: Used to serve files like HTML, CSS, JS.

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

**50. How do you create a REST API in Express.js?**

**1.Set up Express and Routes:**

****

**51 How do you handle session management and authentication in Express.js?**

**1.Using express-session for Sessions:**

****

**2.** **JWT Authentication:**

****

**3.** **Protect Routes using Middleware:**

****

**52 What is dynamic routing in Express.js, and how does it work?**

Dynamic routing means routes can handle variable parameters using :



**53.Why is Express.js good for scalable applications?**

* **Lightweight and Fast**: Minimalistic framework with only core features.
* **Middleware Support**: Easy to add features like authentic and logging.
* **Routing System**: Clean and structured URL handling.
* **Scalability**: Works well in microservices architecture.
* **Easy Integration**: Can use databases like MySQL, MongoDB, etc.
* **Template Engines**: Supports rendering views (EJS, Pug).
* **RESTful API Support**: Ideal for building APIs.

**MYSQL**

**54. What is the difference between CHAR and VARCHAR in MySQL?**

**CHAR(n)**:

Fixed-length string, always occupies n bytes even if the actual data is shorter.

**VARCHAR(n)**:

Variable-length string, only uses as many bytes as needed (plus 1 or 2 extra bytes for length storage).

**55. Explain the difference between INNER JOIN and OUTER JOIN.**

**INNER JOIN**:

Returns only matching rows from both tables.

**OUTER JOIN**:

Returns all rows from one table, even if there’s no match in the other.

**55 What is the difference between WHERE and HAVING clauses?**

**WHERE**: Filters rows **before** grouping. Used with **regular conditions**.

**HAVING**:

Filters rows **after** grouping. Used with **aggregate functions** (SUM, COUNT).

**56. What is an index in MySQL? Why do we use it?**

An **index** speeds up searches by creating a **sorted data structure** for columns. Without indexes, MySQL scans every row (slow).

**Types:**

**Primary Index**: Automatically created on PRIMARY KEY.

**Unique Index**: Prevents duplicates.

**Full-Text Index**: Optimized for searching text.

**57. Which storage engine in MySQL supports foreign keys?**

Only **InnoDB** supports foreign key constraints. **MyISAM does not**.

**58. What does AUTO\_INCREMENT do in MySQL?**

* It automatically **increments** the value of a column for each new row.

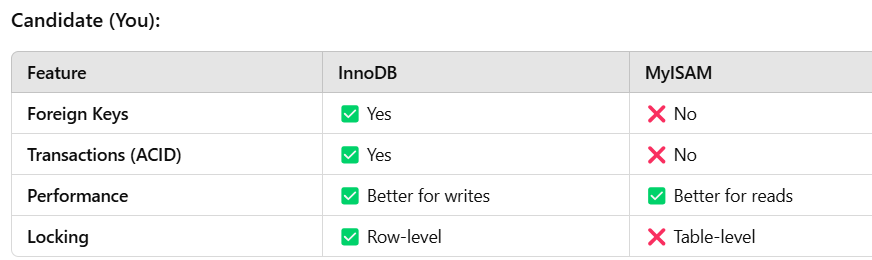
CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50)

);

**59. What is the difference between InnoDB and MyISAM?**

****

**60. What is Normalization?**Normalization is organizing data to reduce redundancy and improve efficiency.

**Forms:**  
1NF: No duplicate columns. 2NF: No partial dependencies.

3NF: No transitive dependencies.

**61. What are ACID properties?**

**1. Atomicity**: Transaction is **all or nothing**.  
2. **Consistency**: Data remains valid before and after transactions.  
3. **Isolation**: Transactions execute **independently**.  
4️. **Durability**: Data is **permanently saved** after a transaction.

62. **What is a Stored Procedure?**

It is a reusable **SQL function** stored in the database.

Faster execution Time Prevents SQL injection. Promotes Reusability.

63.**What is Cascade in MySQL?**

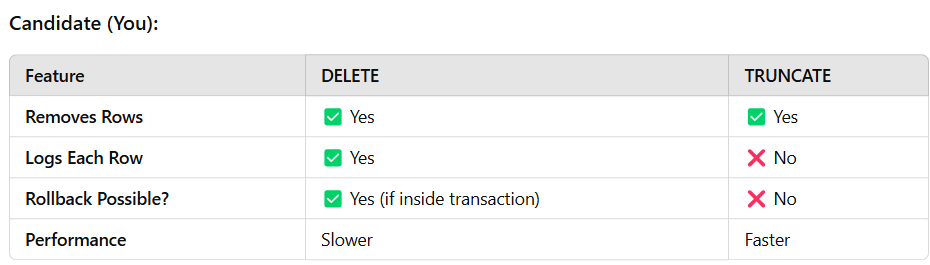
CASCADE automatically **deletes or updates** child records when the parent record changes.

If a referenced user is deleted, their related orders are also deleted.

64. **What is a Full-Text Index?**

* A **full-text index** improves **text searches** in large tables.
* It allows **fast searches in large text fields**.

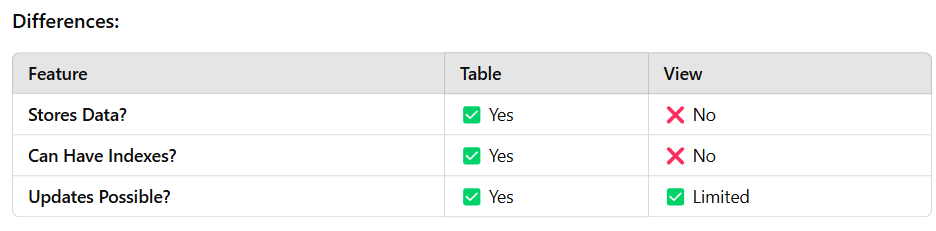
65. How does DELETE differ from TRUNCATE?



**66. What is a foreign key constraint?**  
It enforces **referential integrity**, ensuring valid relationships between tables.

It prevents deleting a user **if there are related orders**.

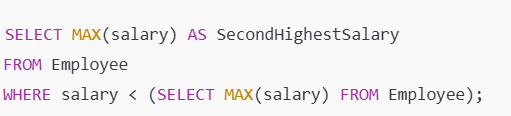
67. **What are views in MySQL?**  
A **view** is a **virtual table** based on a query result. Unlike tables, **views don’t store data**.



**68. What are stored functions in MySQL, and why do we use them?**

* It is a reusable SQL function that returns a value.
* It helps in **code reusability** and **performance optimization**.

69. **How do you find the second-highest salary from an employee table?**

****

**70. How do you count the number of records in a table?**

SELECT COUNT(\*) FROM Employee;

71. **What is the command to create a view?**

CREATE VIEW EmployeeView AS SELECT id, name, salary FROM Employee;

**72. How to remove duplicate values of a column from a result set?**

SELECT DISTINCT column\_name FROM table\_name;

73. **What does the DROP command do in MySQL?**  
DROP permanently **deletes a table, database, or view** from MySQL.

74. **What is the maximum length of VARCHAR?**  
The maximum length of VARCHAR is **65,535** bytes, depending on row size.

75. **Which function returns today’s date and current time?**

SELECT NOW(); -- Returns current date and time

SELECT CURDATE(); -- Returns only the date

SELECT CURTIME(); -- Returns only the time

76. **Which data type can store images in MySQL?**

BLOB (Binary Large Object) is used to store images.

77. **What is the SQL to check the list of tables?** SHOW TABLES;

78. **What is the SQL to execute a stored procedure?**

CALL ProcedureName(parameters);

79. **What is the significance of the HAVING clause in MySQL?**

* HAVING filters records **after aggregation** (GROUP BY).
* WHERE filters **before aggregation**.

80. **What is the SQL to fetch details of all users who registered in 2024?**

SELECT \* FROM Users WHERE YEAR(registration\_date) = 2024;

81. **What type of join returns all records from the left table and matched records from the right table?** LEFT JOIN

82.**Write a query to find the maximum, minimum, and average salary of employees.**

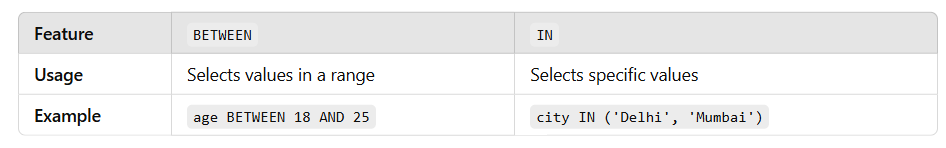
SELECT MAX(salary) AS MaxSalary, MIN(salary) AS MinSalary, AVG(salary) AS AvgSalary FROM Employee;

83. **Write a SQL query to find the total number of orders placed by each customer.**

SELECT customer\_id, COUNT(\*) AS TotalOrders FROM Orders GROUP BY customer\_id;

84.**What is a Composite Key in SQL?**  
It is a **primary key made of multiple columns**.

**85. Explain the difference between BETWEEN and IN in SQL.**

****

**86. What are NULL values in SQL, and how are they handled?**

* NULL represents missing or unknown values.
* Use IS NULL or IS NOT NULL to check for NULL.

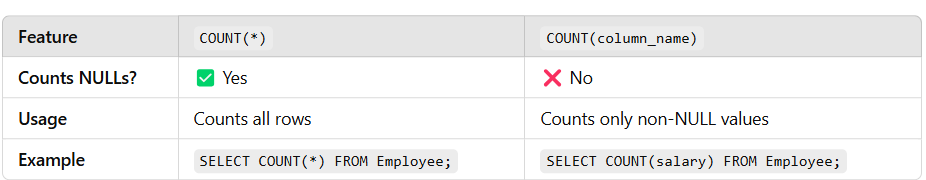
SELECT \* FROM Users WHERE email IS NULL;

87. **What are aggregate functions in SQL? Can you name some?**

They perform calculations on **multiple rows**.

SUM(): Total AVG(): Average COUNT(): Number of rows MAX(): Maximum MIN(): Minimum

88. **Difference between COUNT(\*) and COUNT(column\_name)?**



**87) Difference between git clone and git pull?**

* git clone is used to create a copy of a remote repository to my local machine for the first time.
* git pull is used to fetch the latest changes from the remote repository and merge them into my local branch.

**88) What is git fetch?**

* git fetch downloads the latest changes from the remote repository, but it doesn't automatically merge them into my current branch.
* It updates my local references.

**89) Explain what is a conflict in Git?**

* A conflict happens when two or more people change the same part of a file and Git doesn't know which change to keep.
* It then asks the user to manually resolve it.

**90) What is git add?**

* git add stages the changes I made to files so that they are ready to be committed.
* It tells Git to include updates to a particular file or set of files in the next commit.

**91) What is git commit?**

* git commit captures a snapshot of the currently staged changes in the repository.
* It creates a new commit with a message describing the changes.

**92) What is git push?**

* git push is used to upload my local repository changes to a remote repository like GitHub.

**93) What is a branch in Git?**

* A branch is a separate line of development.
* It allows me to work on different features or bug fixes without affecting the main codebase.

**94) What is git merge?**

* git merge combines the changes from one branch into another, typically merging a feature branch into the main branch.

**95) How will you create a Git repository?**

I can create a Git repository by running git init in the project directory, which initializes an empty Git repository.

**96) What is git stash?**

* git stash temporarily shelves or stores the changes I made to my working directory,
* allowing me to work on something else without committing the unfinished changes.

**97) How to check the log of commits?**

* I can check the commit history using the git log command.
* It shows the list of commits along with the author, date, and message.

**98) Difference between Git and GitHub?**

* Git is a version control system used locally to track code changes, while
* GitHub is a cloud-based hosting service that allows me to store and collaborate on Git repositories online.

**99) How to check the list of branches?**

* I can use git branch to see all local branches.
* To see both local and remote branches, I can use git branch -a.

**100) What is git revert?**

* git revert creates a new commit that undoes the changes of a previous commit without changing the project history.

**101) Which command defines the author email to be used for all commits by the current user?**

git config --global user.email "you@example.com"

**102) What command creates an empty Git repository?**

git init

It initializes a new, empty Git repository.

**Difference between git merge and git rebase**

* git merge combines two branches and creates a new **merge commit**, preserving the history as it happened.
* git rebase moves or re-applies my commits on top of another branch, creating a **linear and cleaner history**.

**Tip**: Use rebase when you want a clean project history, and merge when preserving the full record of changes is important.

**How to resolve a Git conflict**

* First, Git will mark the conflicting files after a pull or merge.
* I open the files, manually choose or combine the changes, then stage the resolved files with git add, and finally complete the process with git commit.  
  git add <conflicted-files> git commit

**How to undo the last commit**

* If I want to undo the last commit but keep the changes in my working directory, I use:

git reset --soft HEAD~1

* If I want to completely remove the last commit and discard the changes, I use:

git reset --hard HEAD~1

* "Git is a distributed version control system that helps developers track changes in their code, collaborate with others, and maintain different versions of a project efficiently.
* It allows us to work offline, manage branches, and safely merge code changes across teams."
* "Git is a tool that manages code changes, supports team collaboration, and keeps the complete project history."

**103) What is TypeScript?**

* TyS is a strongly typed, object-oriented programming language that builds on JavaScript by adding static type definitions.
* It helps catch errors during development and makes large codebases easier to maintain.

**104) What is any in TypeScript?**

* any is a type that disables all type checking for a variable.
* It tells the TypeScript compiler to treat the variable as dynamic, allowing any kind of value to be assigned to it.

**105) How do we define the type of an Array?**

In TypeScript, we can define an array type in two ways:

* Using square brackets, like number[] for an array of numbers.
* Or using a generic type, like Array<number>.

Example: let numbers: number[] = [1, 2, 3];

**106) What is tsconfig.json?**

* It is a configuration file in a TypeScript project that specifies the compiler options and files to be included.
* It helps the TypeScript compiler know how to compile the project.

**107) What are Singleton classes?**

* It is a class that allows only **one instance** of itself to be created during the entire lifetime of the application.
* It is used when exactly one object is needed to coordinate actions across the system.