**What’s New learn JS/nodejs/express js**

* **Morgan** is a popular Node.js middleware that is used for logging HTTP requests and responses in your application. It provides detailed information about incoming HTTP requests, such as the HTTP method, URL, status code, and response time. This can be immensely helpful for debugging, monitoring, and analyzing your web application's traffic.Using morgan('tiny') is helpful when you want a quick overview of your server's request and response activity without overwhelming your logs with too much detail. It's a good choice for development and debugging purposes. You can, however, choose different log formats provided by Morgan to suit your specific logging needs.
* Jwt create openssl rand -base64 32
* **app.disable('x-powered-by')** is a method call commonly used in Express.js applications to disable the X-Powered-By HTTP header that is automatically included in the response headers sent by Express. The X-Powered-By header is an identifying header that typically specifies the technology stack or framework being used to serve the web application. \*\*\*\*\*
* **mongodb-memory-server** is a Node.js library that allows you to run an in-memory MongoDB database for testing and development purposes. It provides a convenient way to create and manage MongoDB instances in memory, without the need to set up a separate MongoDB server or use an external database for your tests. Key features and use cases of mongodb-memory-server include: In-Memory MongoDB: It allows you to create an in-memory MongoDB database that is stored in RAM, making it extremely fast and suitable for testing scenarios where you need a lightweight, temporary database. No Installation Required: You don't need to install MongoDB on your system or set up a MongoDB server. The library manages everything internally.\*\*\*
* In JavaScript, a callback function is a function that is passed as an argument to another function and is intended to be executed after the completion of that function. Callback functions are a fundamental concept in asynchronous programming and are commonly used to handle tasks such as data retrieval, event handling, and asynchronous operations like AJAX requests or timeouts. The primary purpose of callback functions is to ensure that certain code is executed only after a particular task has been completed. This is especially important in JavaScript, which is single-threaded and non-blocking. Here's a basic example of how callback functions work:
* Promises are a fundamental concept in JavaScript that provides a way to work with asynchronous operations in a more structured and predictable manner. Promises are a core part of modern JavaScript and are widely used in web development, especially when dealing with tasks like making network requests, reading files, or querying databases. Here are some key points about Promises: Asynchronous Operations: Promises are designed to handle asynchronous operations, such as fetching data from a server or reading files. They help you write code that doesn't block the main thread, ensuring a more responsive and efficient application. States: Promises have three possible states: Pending: The initial state before the Promise is resolved or rejected. Fulfilled (Resolved): The state when the Promise is successfully completed, typically with a result value. Rejected: The state when the Promise encounters an error or fails, accompanied by a reason for failure.
* Promise.all() is a method used with Promises in JavaScript to handle multiple asynchronous operations in parallel. It takes an array of Promises and returns a new Promise. This new Promise is resolved when all the Promises in the array are resolved, or it is rejected if any one of the Promises is rejected.

const allPromises = Promise.all(promises1,promises2);

Call back function()

* JavaScript, then and catch are methods used with Promises to handle the results and errors of asynchronous operations. They are part of the Promise API and provide a structured way to work with asynchronous code. Here's an explanation of then and catch: then Method: The then method is used to specify what should happen when a Promise is successfully resolved. It takes one or two callback functions as arguments: The first function (often referred to as the "fulfillment handler") is called when the Promise is resolved successfully. It receives the resolved value as an argument. The second function (optional, and often referred to as the "rejection handler") is called when the Promise is rejected. It receives the reason for the rejection as an argument.Callback functions are widely used in JavaScript for event handling, handling responses from server requests, and managing asynchronous code flows. They allow you to write code that can respond to events or data when they become available, without blocking the main thread of execution.

Split(“ ”) function return array.