#Describe the difference between var let and const in javascript.

:- var let and const are keywords in javascript used for declaring variables, but they differ in their scope, hoisting, and reassignment.

Var :

- Variables declared with var are accessible within the function they are declared in, or globally if declared outside any function.
- Variables declared using var are function scoped
- Variables declared using var can be reassigned a new value.
- In this variables are hoisted to the top of their scope and initialized with undefined. This means we can use the variable before its declaration, although it will have the value undefined.

Let :

- Variables declared with let are only accessible within the block they are declared in.
- Variables declared using let are block scoped.
- Variables declared using let can be reassigned a new value.
- In this variables are hoisted to the top of their scope, but not initialized. Attempting to access a let variable before its declaration results in a ReferenceError.

Const :

- Variables declared using const are block scoped, just like in let.
- So variables declared with let are only accessible within the block they are declared in.
- Variables declared using const cannot be reassigned a new value. It must be initialized when declared, and its value cannot be changed later.
- In this variables are hoisted to the top of their scope, but not initialized & just like with let attempting to access a const variable before its declaration results in a ReferenceError.

#How does the javascript event loop works?

:- The JavaScript event loop is a mechanism that manages the execution of code in a non-blocking way.When JavaScript code runs, functions are added to the call stack, which is a LIFO (Last In, First Out) data structure.

Also, when an asynchronous operation (like a timer or network request) is encountered, it's handed off to the browser's API. The asynchronous operation completes, its callback function is added to the callback queue.

The event loop continuously checks if the call stack is empty. If it is, it takes the first callback from the callback queue and pushes it onto the call stack for execution.

This way, JavaScript can handle multiple tasks concurrently without blocking the main thread.

#In Node.js what is the purpose of module.export?

:- module.exports is the object that is returned by the require() function when you import a module. It allows you to expose variables, functions, classes, or any other JavaScript entity from one module to be used in another.