- Nodejs is single threaded. It means that each process has only one thread of execution.

→ Nodejs uses Non-blocking I/O

Non-blocking means that we can make a request while doing something else and then, when this request is finished, a callback will execute to handle the result.

→ Asynchronous programming

Asynchronous programming is a key feature of Node JS. Node JS provides several mechanisms for handling asynchronous code, including callbacks, promises, and async/await.

## ← Module Pattern In Node JS →

* **Module :—**

Module is a mechanism for splitting JS programs into separate manageable chunk called as module, that can be imported whenever needed.

There are two ways for implementing modules :—

1. CJS (common JS module) → Express
2. ESM (ES6 modules) → React

## * module.exports :-

In Node.js, 'module.exports' is an object that is used to define what a module exports and makes available for other module to use.

When we create a module in Node.js, any variable, functions, or objects that we want to expose to other modules must be added to the 'module.exports' object.

For example, we have a file called 'myModule.js' and we want to export a function called 'addNumbers':

```
function addNumbers(a, b) {
    return a + b;
}
module.exports = { addNumbers };
```

This above code means that when we import 'myModule.js' into another file using the 'require()' function, we can access the 'addNumbers' function like this.

```
const myModule = require('./myModule');
const sum = myModule.addNumbers(2, 3);
console.log(sum);  // output: 5
```

By default 'module.exports' is an empty object, but we can assign any value to it - an object, a function, or a primitive value, such as a string or number.

* Difference between 'named export' and 'default export' and '* as export';

⇒ **named export**: name export allow a module to export multiple values, each with a unique name. These can be imported into another module using same name.
For example:

```
// File: math.js
export const sum = (a, b) => a + b;
export const multiply = (a, b) => a * b;

// File: main.js
import { sum, multiply } from "./math.js";
console.log(sum(2, 3)); // 5
console.log(multiply(2, 3)); // 6
```

⇒ **Default export**:-
It allows a module to export a single value as the default export. This value can be imported into another module using any name. For example.

```
// File: math.js
const sum = (a, b) => a + b;
export default sum;

// File: main.js
import mysum from "./math.js";
console.log(mysum(2, 3)); // 5
```

Date: / /

⇒ "* as" :-

This syntax allows to import all the exports from a module and bind them to a specific object.

```
// File: math.js
export const sum = (a, b) => a+b;
export const multiply = (a, b) => a*b;

// File: main.js
import * as math from './math.js';
console.log(math.sum(2, 3)); // 5
console.log(math.multiply(2, 3)); // 6
```