Two Module Systems Exist

Feature	ES Modules (ESM)	Node.js CommonJS (CJS)
Syntax	import / export	require() / module.exports
Introduced	Modern JavaScript	Old Node.js system
Enabled by	"type": "module" in package.json	Default behavior in Node
File extension	.mjs or .js (with type:module)	.js
Execution time	Static (checked before run)	Dynamic (runtime)

■ ES Modules (import/export)
Used in modern JS + Express (new way)

Export
```
export function home() {}
export const name = "Malaika";
export default function(){};
```

Import
```
import home, { name } from "./file.js";
```

■ Supports default + named exports
■ Faster because Node analyzes imports before running code
■ Works like browser JavaScript
■ Must use "type":"module" in package.json

■ CommonJS (Node.js default way)

Export
```
module.exports = {
home: function(){},
name: "Malaika"
};
```

Or default style:
```
module.exports = function() {};
```

Import
```
const { home, name } = require("./file");
```

■ Default in Node.js
■ Good for older projects
■ No native export default — you simulate it
■ Dynamic, slower in large apps

■ Key Difference in Export Style

Concept	ES Modules	CommonJS
Default Export	export default func	module.exports = func
Named Export	export { func }	exports.func = func
Import Default	import func from	const func = require()
Import Named	import { func } from	const { func } = require()
Timing	Static (compile time)	Dynamic (runtime)

■ Real Example Comparison

■ ESM
home.js
```
export default function home() {
```

```
return "Home Page";
}
export function contact() {
return "Contact Page";
}
```

server.js

```
import express from "express";
import home, { contact } from "./home.js";
```

■ CommonJS

home.js

```
function home() {
return "Home Page";
}
function contact() {
return "Contact Page";
}
module.exports = { home, contact };
```

server.js

```
const express = require("express");
const { home, contact } = require("./home");
```

■ Easy Memory Trick

If you say... Use
"I want modern JS like React" ■ ES Modules
"My project is old Node" ■ CommonJS

■ Final Answer (Simple Words)

ES Modules uses import/export — modern, faster, browser-friendly.
Node.js modules use require/module.exports — old system, still supported.

You are learning Express — so prefer ES Modules now ■