



[linkedin.com/in/maortizolid](https://www.linkedin.com/in/maortizolid)



[maortizolid@gmail.com](mailto:maortizolid@gmail.com)

# Módulos en NodeJS

El sistema de módulos en Node.js es un mecanismo que permite organizar y reutilizar el código en aplicaciones JavaScript. Hay dos sistemas principales de módulos en Node.js: **CommonJS** (CJS) y **ECMAScript Modules** (ESM).

Una de las diferencias más importantes entre CommonJS (CJS) y ECMAScript Modules (ESM) es el enfoque de importación:

- **CommonJS (Importación Estática):**
  - Las importaciones se resuelven durante la fase de compilación.
  - Las dependencias se cargan sincrónicamente.
  - Sintaxis de importación con `require`.
- **ESM (Importación Dinámica):**
  - Permite importación dinámica durante la ejecución.
  - Las dependencias pueden cargarse asincrónicamente.
  - Sintaxis de importación con `import`.

## CommonJS (CJS)

- Utilizado por defecto en versiones antiguas de Node.js.
- Sintaxis de Importación: `const modulo = require('nombre_del_modulo');`
- Sintaxis de Exportación: `module.exports = valor;`
- **Ventajas:**
  - *Amplia Adopción:* CommonJS es ampliamente adoptado y utilizado en la comunidad de Node.js, lo que significa que hay una gran cantidad de módulos disponibles en este formato.
  - *Sintaxis Simple:* La sintaxis de importación y exportación es sencilla y fácil de entender.
  - *Compatibilidad con Versiones Antiguas:* Es compatible con versiones más antiguas de Node.js, lo que puede ser crucial en algunos entornos.
- **Desventajas:**
  - *Carga Sincrónica:* CommonJS carga módulos de forma síncrona, lo que puede afectar el rendimiento en aplicaciones grandes o en situaciones donde la carga asíncrona sería más eficiente.
  - *Limitaciones de Importación:* La importación es estática y no admite algunas características más avanzadas, como la importación dinámica.

## ECMAScript Modules (ESM)

- Introducido en versiones más recientes de Node.js (a partir de la versión 13) y compatible con navegadores modernos.
- Sintaxis de Importación: `import modulo from 'nombre_del_modulo';`
- Sintaxis de Exportación: `export default valor;`
- **Ventajas:**
  - *Sintaxis Moderna:* ESM utiliza una sintaxis de importación/exportación más moderna y concisa, similar a la de otros lenguajes de programación.
  - *Carga Asíncronica:* ESM permite la carga asíncronica de módulos, lo que puede mejorar el rendimiento en situaciones específicas.
  - *Importación Dinámica:* Admite la importación dinámica, lo que facilita la carga de módulos de forma condicional o dinámica en tiempo de ejecución.
- **Desventajas:**
  - *Compatibilidad:* Puede requerir configuración adicional en entornos más antiguos de Node.js, y no es compatible directamente con la mayoría de los módulos CommonJS existentes.
  - *Curva de Aprendizaje:* Puede haber una curva de aprendizaje para quienes están acostumbrados a CommonJS.

## Ejercicios

Vamos a ver unos sencillos ejemplos de cómo utilizar estos dos sistemas de módulos de NodeJS.

### CommonJS (CJS)

Creamos una carpeta y dentro de ella el archivo `math.js` y escribimos el siguiente código:

```
function sum (a, b) {  
  return a + b  
}
```

```
// CommonJS module export  
module.exports = sum
```

Declaramos una función que recibe dos números y devuelve la suma. Exportamos dicha función con `module.exports`

Ahora creamos el archivo `index.js` y escribimos el siguiente código:

```
// CommonJS require module
const suma = require('./math')

console.log(suma(1, 2))
```

Importamos el archivo `math.js` y ya podemos utilizarlo en este archivo llamando a la función exportada `sum`.

## ES Modules (ESM)

Creamos otra carpeta carpeta y dentro de ella el archivo `math.mjs` y escribimos el siguiente código:

```
// ES Modules export
export function sum (a, b) {
  return a + b
}
```

Directamente exportamos la función con `export`.

Ahora creamos el archivo `index.mjs` y escribimos el siguiente código:

```
// ES Modules import
import { sum } from './math.mjs'

console.log(sum(1, 2))
```

Importamos el archivo `math.mjs` y ya podemos utilizarlo en este archivo llamando a la función exportada `sum`.

Como podéis observar, en el caso de utilizar ES Modules, hemos utilizado la extensión `.mjs` para ambos archivos. Esto nos permite utilizar el sistema de módulos ECMAScript.

Por defecto la extensión `.js` utiliza el sistema de módulos CommonJS, aunque si queremos forzarlo a utilizar este sistema de módulos, podemos utilizar la extensión `.cjs`

- `.js` -> por defecto utiliza CommonJS
- `.mjs` -> para utilizar ES Modules
- `.cjs` -> para utilizar CommonJS

Más adelante cuando veamos y analizemos el archivo `package.json`, veremos como configurarlo para utilizar el sistema de módulos ECMAScript (ESM) utilizando la extensión `.js`