Módulos en JavaScript

Sebastián Forero Duque

Santiago Gallego Henao

En JavaScript, los módulos se utilizan para organizar el código en bloques manejables, lo que es fundamental para gestionar la complejidad del software. Son el mecanismo de alto nivel para lograr esto en Node.

Importación de Módulos

Es común importar módulos para utilizar funcionalidades definidas por otros o código de tu propio proyecto:

Desde librerías del sistema de archivos (fs):

```
import fs from 'fs/promises';
```

Desde archivos locales del proyecto:

```
import utils from './utils/mjs';
```

Importación de partes específicas de un módulo (si fueron exportadas con nombre):

```
import { specificUtil } from
'./utils.mjs';
```

Exportación de Módulos

La exportación de módulos se realiza de las siguientes maneras:

Exportaciones con nombre:

```
export const utilA = () => { /* ... _/ }; export function utilB() { /_ ... */ };
```

Exportación por defecto:

```
const defaultValue = { /* ... */ }; export default defaultValue;
```

La exportación por defecto es lo que se obtiene con una importación directa, mientras que las exportaciones con nombre se asignan a las importaciones con nombre.

Consideraciones sobre la Extensión de Archivo

El uso de la extensión .mjs en los archivos de módulo es necesario a menos que se establezca "type": "modules" en el archivo package.json de NPM.

Sintaxis de Módulos (ESM vs. CommonJS)

Lo que hemos visto hasta ahora es la sintaxis **ESM (EcmaScript Module Syntax)**, que es el estándar general en Node y JavaScript. Sin embargo, existe otra sintaxis, **CommonJS**, que ha sido utilizada por gran parte de Node durante años y que a menudo verás en el código:

Importación en CommonJS:

```
const fs = require('fs'); const myUtil = require('./utils.js');
```

Exportación en CommonJS:

```
function utilityOne() { /* ... */ } module.exports = { utilityOne }; exports.utilityOne = utilityOne;
```

Diferencias Clave entre ESM y CommonJS

Sintaxis

CommonJS: Utiliza `require()` para importar y `module.exports` para exportar, siendo la sintaxis más antigua.

ESM: Emplea `import` y `export`, representando la sintaxis moderna y estandarizada de JavaScript.

Archivos y Configuración

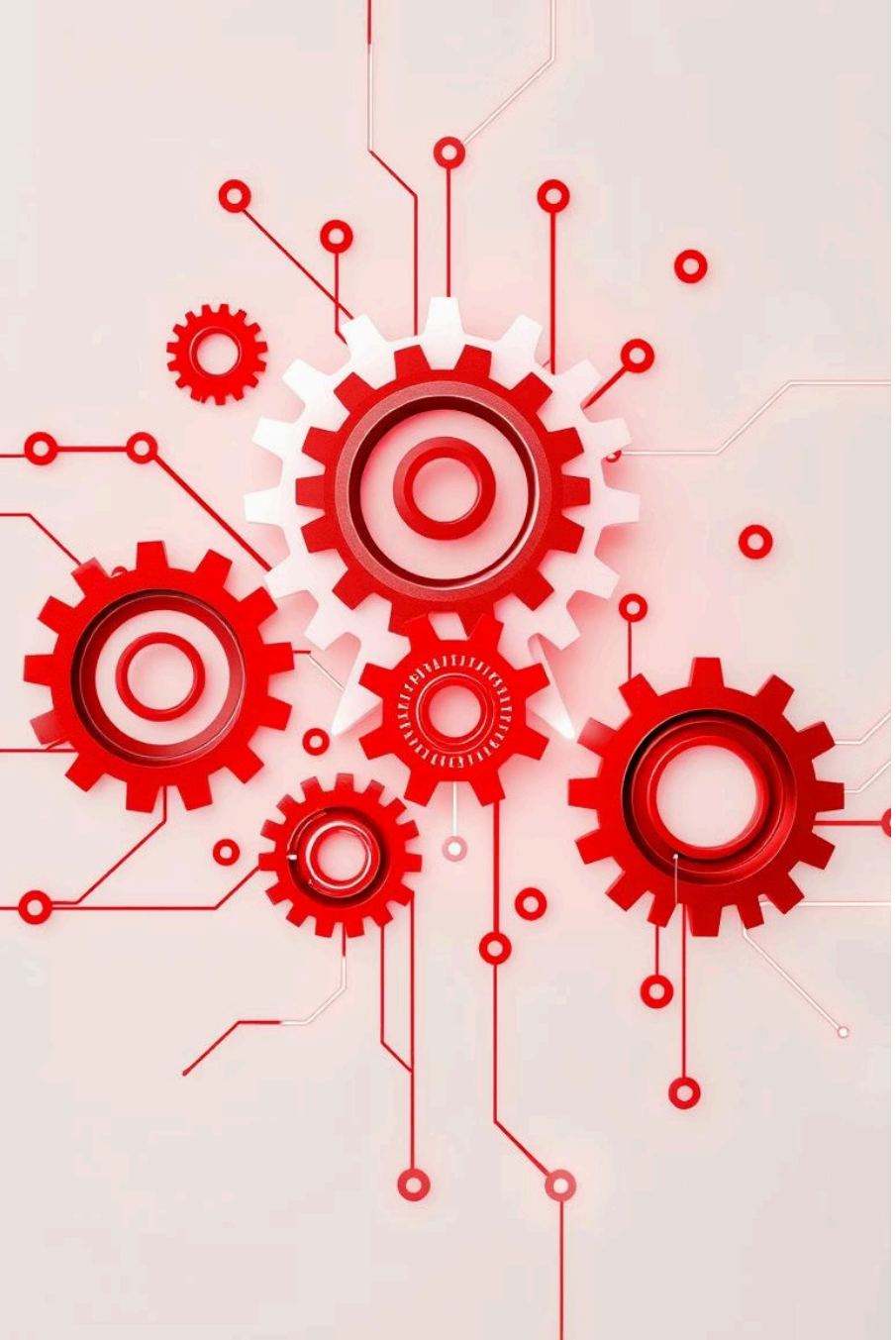
ESM: Requiere el uso de la extensión `.js` en las importaciones de módulos y, para Node.js, necesita que `"type": "module"` se establezca en el `package.json`.

CommonJS: Generalmente no requiere configuraciones específicas ni extensiones especiales más allá del `.js` por defecto.

Flexibilidad de Exportación

ESM: Soporta de manera más elegante tanto las exportaciones nombradas (`export const name`) como las exportaciones por defecto (`export default value`), facilitando la estructuración del código.

CommonJS: Principalmente utiliza `module.exports` para exportar un solo valor o un objeto con múltiples propiedades.



En resumen

En resumen, la clave más importante de esta sección es comprender cómo la división del software en componentes es absolutamente fundamental para gestionar la complejidad, y los módulos son el mecanismo principal para lograrlo en Node.

Referencias

https://www.infoworld.com/article/2252306/10-javascript-concepts-every-nodejs-developer-must-master.html