Il file system

Node, essendo orientato allo sviluppo lato server, deve poter leggere, scrivere, modificare e cancellare file, per fare ciò mette a disposizione un modulo per l'accesso al file system chiamato **fs**.

Questo modulo non ha bisogno di essere installato: basta importarlo con **require('fs')** (o con **import fs from 'fs'** se usi i moduli ECMAScript)

Asyncrono e sincrono

Una caratteristica interessante è che la maggior parte delle funzioni del modulo fs sono disponibili in due varianti:

- una sincrona (di solito terminano con “Sync”)
- una asincrona

L'approccio asincrono è più in linea con la filosofia di Node.js, che punta all'efficienza e alla non-bloccabilità del thread principale.

Asyncrono e sincrono

Nel contesto di un'applicazione Node, la gestione del filesystem è utile in molte situazioni:

- salvare log
- gestire file di configurazione
- caricare template o dati statici
- permettere agli utenti di caricare file (ad es. immagini)
- manipolare file temporanei

È importante ricordare che lavorare direttamente con il filesystem può introdurre problemi di concorrenza, permessi o accesso a file indesiderati. Per questo è buona pratica validare i percorsi e gestire con attenzione gli errori.

fs.promises

Dalla versione 10 in poi, Node ha introdotto anche un'interfaccia basata su Promises (fs.promises).

Questo rende molto più semplice usare async/await, migliorando la leggibilità del codice asincrono.

```
const fs = require('fs/promises');

async function leggiFile() {
  try {
    const contenuto = await fs.readFile('testo.txt', 'utf8');
    console.log(contenuto);
  } catch (err) {
    console.error('Errore:', err);
  }
}
```

Metodi di fs

Metodo	Scopo principale	Variante sincrona	Note principali
`fs.mkdir`	Crea una nuova cartella	**`fs.mkdirSync`**	Può creare cartelle annidate con `{ recursive: true }`
`fs.rm`	Rimuove file o cartelle	**`fs.rmSync`**	Usa `{ recursive: true }` per eliminare intere directory
`fs.readdir`	Legge il contenuto di una directory	**`fs.readdirSync`**	Restituisce un array di nomi di file/cartelle
`fs.access`	Controlla se un file o cartella esiste	**`fs.accessSync`**	Utile per verificare permessi o presenza
`fs.stat`	Restituisce informazioni su file/cartella	**`fs.statSync`**	Fornisce dati come dimensione, tipo, date, ecc.

__filename e __dirname

__filename e __dirname sono due costanti globali disponibili in ogni modulo. Servono a sapere dove si trova fisicamente il file che stai eseguendo.

Utili per evitare errori relativi a **process.cwd()** (che cambia a seconda di dove lanci Node)

Attenzione: Se stai usando i moduli ECMAScript (ESM), __filename e __dirname non esistono. Devi ricostruirli manualmente (esempio nella slide successiva)

```
import { fileURLToPath } from 'url';
import path from 'path';

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

console.log('__filename:', __filename);
console.log('__dirname:', __dirname);
```