

NODEJS STANDARD LIBRARY

INTRODUZIONE A NODEJS

▶ **Visual Studio Code**

▶ **Git**

▶ **NodeJS**

- ▶ Javascript è un linguaggio interpretato; ogni browser include un interprete JS per l'esecuzione di codice nelle pagine web.
 - ▶ Firefox -> SpiderMonkey
 - ▶ Edge (vecchia versione) -> Chakra
 - ▶ Chrome -> V8
- ▶ NodeJS estrapola V8 dal contesto del browser web e crea un ambiente runtime per l'esecuzione di codice JS in ambito Desktop/Server.



- ▶ Si può usare un linguaggio storicamente limitato al frontend anche per lo sviluppo Desktop/Backend
- ▶ Tutte le API standard utilizzabili nei browser rimangono valide (e.g. Array, Promise, ArrayBuffer etc.)
- ▶ Le API standard vengono arricchite da una libreria standard che consente accesso alla rete, al file system, alle variabili d'ambiente etc.
- ▶ Essendo V8 scritto in C++ è possibile creare moduli NodeJS anche in C++, potendo così sfruttare tutte le potenzialità di un linguaggio «system level»

- ▶ Come tutti i linguaggi di programmazione NodeJS offre una sua libreria standard. La documentazione per l'attuale versione LTS (12.19.0) è consultabile all'indirizzo <https://nodejs.org/dist/latest-v12.x/docs/api/>
- ▶ La libreria standard è organizzata in moduli: alcuni sono ampiamente utilizzati mentre altri non sono pensati per un utilizzo diretto da parte degli sviluppatori (anche se le funzionalità sono esposte e disponibili)
- ▶ Ogni modulo è contrassegnato da uno «stability index» che ne identifica la stabilità. Gli indici possibili sono:
 - ▶ **Stability: 0** — Modulo deprecato. Le funzionalità non sono garantite e possono generare warnings
 - ▶ **Stability: 1** — Sperimentale. Modulo sottoposto a revisioni che non garantiscono la backward-compatibility
 - ▶ **Stability: 2** — Stabile. Totalmente compatibile con l'ecosistema npm e stabilmente mantenuto

- ▶ **Buffer** → Manipolazione di sequenze di byte di lunghezza fissa.
- ▶ **Console** → Debugging. Include i metodi *console.log()*, *console.warn()* e *console.error()*
- ▶ **Crypto** → Fornisce implementazioni di metodi di crittografia (hash, cifrari, firma etc.)
- ▶ **DNS** → Metodi di alto livello per effettuare query DNS
- ▶ **Errors** → Creazione e gestione di errori
- ▶ **Events** → Creazione e gestione di eventi
- ▶ **FS** → Interazione con il file system

- ▶ **HTTP/HTTPS** → Esecuzione di richieste HTTP(S) e creazione di server HTTP(S)
- ▶ **OS** → Utilità e proprietà esposte dal sistema operativo
- ▶ **Path** → Utilità per la manipolazione di percorsi di file e directory
- ▶ **Stream** → Utilità per la manipolazione di flussi di dati provenienti da svariate sorgenti (e.g. richieste HTTP, stdin etc.)