JavaScript Node.js

Moduli

Url: https://github.com/obiSerra/laboratorio-nodejs-2022

Breve riepilogo

- NodeJs è un runtime che permette di eseguire JavaScript “lato server”
- Utilizza V8 come alcuni browser per interpretare JS
- Npm è il packet manager default di Node; fornisce pacchetti e una CLI
- Git è un file versioning tool

Git commands cheat-sheet

1. Preparare i file per un commit (stage file)

   $ git add <file-da-aggiungere> (oppure . per aggiungere tutto)

   (stage file a un editor)

2. Creare un commit

   $ git commit -m “messaggio del commit”

3. Salvare il commit su remot

   $ git push

I moduli in Node

- Single-responsibility-principle (SRP)
- E’ possibile (e consigliato) creare tanti piccoli moduli
- Esistono alcuni moduli “di base” che permettono di interagire con l’ambiente circostante (fs, path, ...)
- Npm e’ il piu’ grande repository al mondo di moduli
- Attenzione a non abusare di tutto questo potere (non usiamo moduli per cose triviali) *(https://www.sciencealert.com/how-a-programmer-almost-broke-the-internet-by-deleting-11-lines-of-code)*

CommonJs

- Metodo standard di NodeJs (ma non di JavaScript frontend)
- require e module.exports (o solamente exports)
- Puo’ essere usato in qualunque punto della pagina
- Carica le dipendenze man mano che trova i require
- Non e’ necessaria l’estensione del file

(ref lezione2/commonjs/src/index.js)

ESM (ES6)

- Metodo di standard del frontend
- Attivabile tramite l’attributo “type” di un package.json ("type":"module")
- import e export (o export default)
- Deve essere usato a inizio pagina
- Carica prima tutte le dipendenze e poi esegue il file
- Necessaria l’estensione del file

(ref lezione2/esm/src/index.js)

Pattern per creare un modulo

- Named Exports *(lezione2/commonjs/src/namedExport.js)*
- Substack pattern (o function exporting) *(lezione2/commonjs/src/substackExport.js)*
- Class export *(lezione2/commonjs/src/classExportModule.js)*
- Instance export *(lezione2/commonjs/src/classExportModule.js)insta*

Object destructuring

In JavaScript e' possibile "destrutturare" un oggetto o un array assegnandone direttamente alcune proprieta' ad alcune variabili:

Dato un oggetto tipo:

```
const someObj = { hello: "world", love: "JavaScript" };
```

Fare

```
const { hello, love } = someObj;
```

E' equivalente a:

```
const hello = someObj.hello;
const love = someObj.love;
```

(notare le graffe per destrutturare un oggetto)

Array destructuring

Lo stesso vale per gli array, ma le proprieta' vengono prese in base all'ordine e non al nome delle proprieta' stesse:

```
const someAr = [1, 2];
```

Questo

```
const [a, b] = someAr;
```

Equivale a questo:

```
const a = someAr[0];
const b = someAr[1];
```

(notare le parentesi quadre per destrutturare un array)

Esercitazione

Requirements:

- scrivete un piccolo programma che tramite un modulo, scriva sulla console utilizzando dei prefissi e suffissi diversi:

es:

writer(“nodejs”) -> nodejs!!!

writer.hello(“roberto”) -> hello roberto

writer.byebye(“roberto”) -> bye roberto, see you later

Moduli di default

Node mette a disposizione diversi moduli di default

- fs - interagire con filesystem *https://nodejs.org/api/fs.html*
- path - gestione delle path *https://nodejs.org/api/path.html*
- os - sistema operativo *https://nodejs.org/api/os.html*
- url - interazione con le URL *https://nodejs.org/api/url.html*
- process - interazione con il processo di node *https://nodejs.org/api/process.html*

Facciamo una prova

Vediamo come scrive una piccola applicazione che legge una pagina di wikipedia e la salva sul nostro computer.

(potete trovare una versione in *lezione2/defaultModules/index.js)*