L’installazione di TypeScript

Docente
Mario Perna
prof.perna.mario@darzo.net

A.S.
2025/2026
Materia
TEPSIT (Laboratorio)

Ambiente di sviluppo

Negli episodi precedenti abbiamo introdotto il funzionamento del linguaggio typescript.

Per poter iniziare a programmare in typescript, ci servirà:

- Un editor di testo (Visual Studio Code)
- NodeJS (preferibilmente una delle ultime versioni disponibili)
- Utilizzo di strumenti come TypeScript Compiler (TSC) e Webpack.

Installazione di Node.js

Per trasformare TypeScript in JavaScript si usa il **TypeScript Compiler** (tsc).
tsc è un pacchetto distribuito tramite **npm** (Node Package Manager).
npm fa parte di **Node.js**, quindi per installare tsc serve avere Node.js già installato.

https://nodejs.org

Installazione di TypeScript

Per installare TypeScript esistono due strade, che sono:

- Installazione globale
- Installazione locale (a livello di progetto)

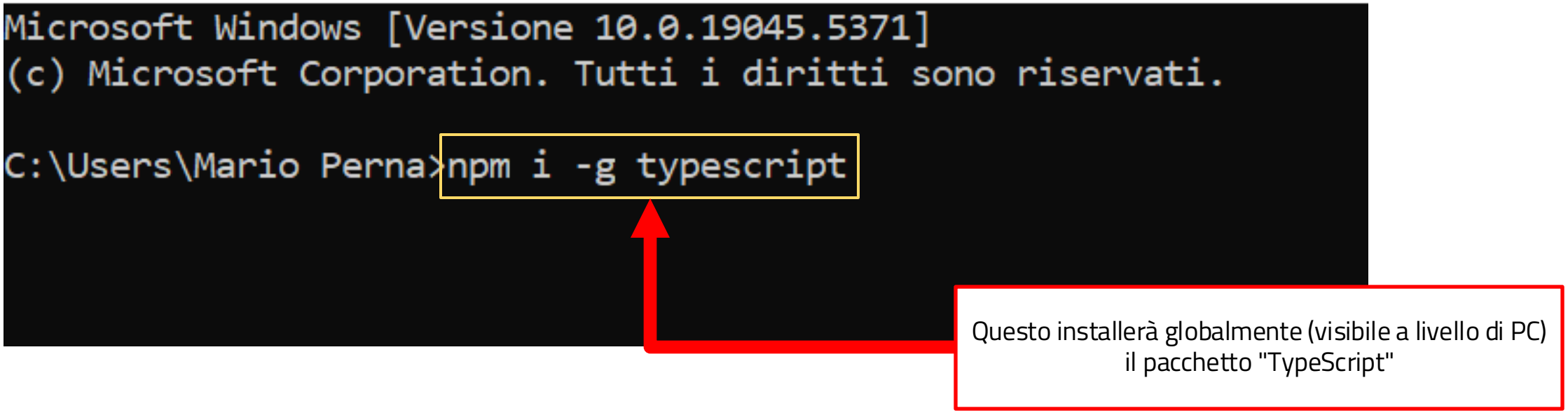
Nelle prossime slides approfondiremo entrambe le installazioni.

Installazione a livello globale

Installazione di TypeScript (Globale)

Creare, alla posizione personale della cartella di rete, una cartella denominata TypeScript. Successivamente, aprire la cartella con Visual Studio Code e dal terminale integrato digitare:

npm i –g typescript



Problemi con ExecutionPolicy?!

Se doveste aver problemi con l'ExecutionPolicy digitare questo comando da PowerShell:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned

Cosa fa nel dettaglio:
-Scope CurrentUser → applica la modifica **solo al tuo utente**, non a tutto il sistema
-ExecutionPolicy RemoteSigned → consente l'esecuzione:

- di **script locali** (anche se non firmati);
- di **script scaricati da Internet**, *solo se firmati digitalmente*.

Dipendenze a livello globale

I pacchetti globali vengono salvati in una directory comune, diversa da quella del progetto.

- Su **Linux/macOS**: /usr/local/lib/node_modules/ (binari in /usr/local/bin/).
- Su **Windows**: C:\Users\<nome>\AppData\Roaming\npm\node_modules\ (binari in …\npm\).

Comandi utili: npm root -g mostra la cartella moduli, npm bin -g quella dei binari.

Installazione a livello di progetto

Installazione di TypeScript (a livello di progetto) – Inizializzazione NPM

Inizializza il progetto per poter utilizzare npm tramite **npm init**

Questo crea il file **package.json,** tramite setup guidato, con le informazioni base del progetto Node.js (nome, versione, dipendenze, script).

```
marioperna@Mac Project1 % npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
```

Installazione di TypeScript (a livello di progetto)

Talvolta non è possibile usare un'unica versione di **TypeScript**, poiché progetti diversi richiedono versioni specifiche.

In questi casi, è consigliabile effettuare un'**installazione locale** di TypeScript all'interno del singolo progetto.

```
marioperna@Mac Project1 % npm i typescript

added 1 package, and audited 2 packages in 737ms

found 0 vulnerabilities
```

Il pacchetto verrà installato solo nella cartella del progetto **Project1**, rendendo i comandi e la visibilità **limitati al progetto stesso**.

Dipendenze a livello di progetto

Quando npm installa le dipendenze, queste vengono salvate nella cartella **node_modules**.

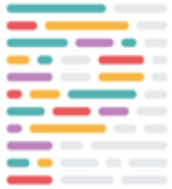
È buona norma **non includere questa cartella** quando si condivide il progetto: il destinatario potrà ricrearla eseguendo **npm i**.

Le estensioni

Estensioni

Alcuni plugin consigliati per lavorare con TypeScript sono:

- **TypeScript**: Fornisce funzionalità di base per il supporto di TypeScript in VS Code, inclusa la segnalazione degli errori e l'autocompletamento del codice.
- **Prettier - Code formatter**: uno dei formattatori di codice più popolari e supporta TypeScript. Questa estensione ti permette di formattare automaticamente il tuo codice TypeScript secondo le convenzioni.



Prettier

Il file tsconfig.json

Impostiamo il file **tsconfig.json**

Il file **tsconfig.json** è il cuore della configurazione di un progetto **TypeScript**.

Definisce **come il compilatore (tsc)** deve tradurre il codice TypeScript in JavaScript, specificando opzioni come:

- la **versione di JavaScript** di destinazione,
- le **cartelle di input/output**,
- le **regole di compilazione** (strict mode, moduli, JSX, ecc.).

```jsonc
{
  "compilerOptions": {
    /* === Target ECMAScript e librerie === */
    "target": "es2015",
    "lib": ["es2015", "dom"],

    /* === Modulo e output === */
    "module": "commonjs",
    "outDir": "./dist",
    "rootDir": "./",

    /* === Opzioni di controllo === */
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true
  },

  /* === Inclusione dei file TypeScript === */
  "include": ["**/*.ts"],

  /* === Esclusione di cartelle inutili === */
  "exclude": ["node_modules", "dist"]
}
```

Siamo pronti... proviamo TypeScript in azione!

Proviamo TypeScript

Creiamo una cartella rinominata **01_intro:** al suo interno creiamo un file rinominato **index.html**

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Primo esercizio</title>
7   </head>
8   <body>
9       <!--Inclusione dello script 'script.ts'-->
10      <script src="script.js"></script>
11  </body>
12  </html>
```

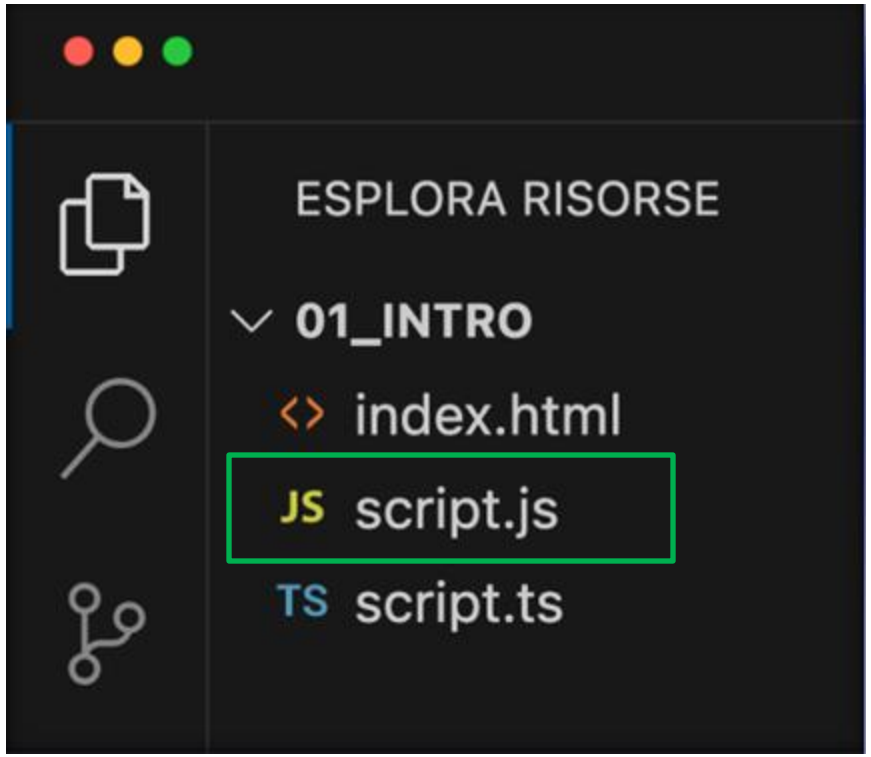
Proviamo TypeScript

Creiamo il file **script.ts**

```
1   function stringaVuota(testo:string):boolean{
2       return testo.length === 0;
3   }
4   const stringa1:string="";
5   const stringa2:string="Ciao mondo";
6
7   document.write("La stringa "+ stringa1+" è vuota? "+stringaVuota(stringa1));
8   document.write("<br/>");
9   document.write("La stringa "+stringa2+" è vuota?"+stringaVuota(stringa1));
```

Compiliamo il file TypeScript (con installazione globale)

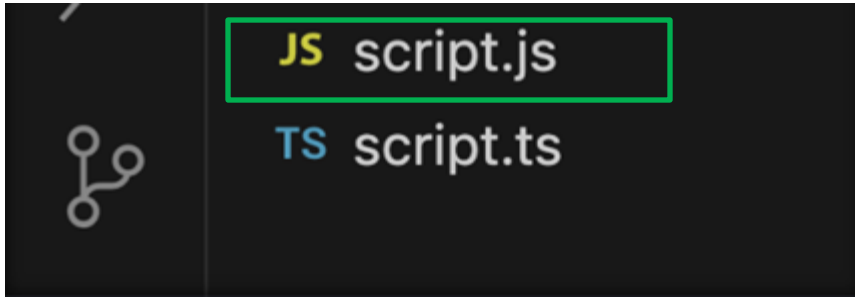
Da terminale digitiamo **tsc script.ts** A questo punto nella cartella del progetto viene creato il file **script.js**



Compiliamo il file TypeScript (con installazione dipendenze a livello di progetto)

Da terminale digitiamo **npx tsc script.ts** A questo punto nella cartella del progetto viene creato il file **script.js**

```
marioperna@Mac Project1 % npx tsc main.ts
```



Errore “Duplicate Variable Block Scope” in TypeScript

L’errore “Duplicate Variable Block Scope” indica che una variabile è dichiarata più volte nello stesso ambito.
Questo errore può derivare da file .js e .ts duplicati o da una configurazione mancante di **tsconfig.json**.

Assicurarsi di escludere i file compilati e di evitare doppie dichiarazioni.

```
{
  "exclude": ["node_modules", "**/*.js"]
}
```

Compiliamo il file TypeScript

Questo è il risultato finale:

```
La stringa è vuota? true
La stringa Ciao mondo è vuota?true
```

Se effettuate delle modifiche al codice TS è richiesta nuovamente la compilazione del file mediante il comando tsc

Sitografia

https://it.javascript.info/
https://www.w3schools.com/js/default.asp