TYPESCRIPT

WHAT'S TYPESCRIPT: PUNTI DI FORZA E DIFFERENZE CON JS VANILLA

- Super-set di Javascript
- Static typing VS Dynamic typing
- Forza a scrivere codice pulito dall'inizio
- L'IDE evidenzia errori

IL COMPILATORE

- Il browser NON legge typescript
- Typescript va compilato in JS

IL COMPILATORE

Nel terminale:

- npm i -g typescript. Installa TS.
- tsc - -init. Crea il JSON di compilazione.
- tsc <filePathAndName>. Compila TS in JS
- tsc <filePathAndName> -w . Compila ad ogni modifica

TYPES: TYPE INFERENCE

- Tipizzazione implicita in fase di dichiarazione di variabile o costante.
- E.g. *let myVar = 3* implica *let myVar: number = 3*

TYPES: TYPE ANY

- Rende di fatto possibile la tipizzazione dinamica
  - *let myVar: any = ‘Hello’*
  - *myVar = 5*

L’assegnazione di un valore di tipo differente rispetto a quello stabilito in fase di dichiarazione, è possibile solo con il type any

TYPES: TYPE UNION

- Rende possibile l’assegnazione di valori di tipo diverso, stabiliti al momento della dichiarazione
  - *let myVar: number | string;*
  - *myVar = 5*
  - *myVar = ‘Hello’*

TYPES: TYPE UNION ARRAY

- La tipizzazione vale anche per gli array
  - *let arrString: string [] = [‘Hello’, ‘World’].* Array di string
  - *let arrNumbers: number [] = [1, 2, 3].* Array di number
  - *Let arrMix: (string | number) [] = [1, ‘Hello’].* Array di string e number
- Così come la type inference
  - *let arrMix [] = [1, ‘Hello’, true].* Array di number, string, boolean

TYPES: TUPLES

- Array che prende solo determinati tipi in un ordine predefinito
  - *let arrTup: [string, number] = [‘Hello’, 5]*
  - *arrTup = [4, ‘Hi’].* **ERROR. IL COMPILATORE DA ERRORE**
  - *arrTup = [‘Ciao’, 23].* **OK**

INTERFACES: UN MODELLO PER OGGETTI

- Un oggetto costruito tramite una interface DEVE aderire al modello.
- Le coppie *key: value* vengono stabilite attraverso una sintassi *key: type*
  - *interface myInterface { name: string; age: number: }*
  - *let myObj: myInterface = { name: 'Marco', age: '27' }* **ERROR**
  - *let myObj: myInterface = { name: 'Marco', age: 27 }* **OK**

TYPES VS INTERFACES

- Una interface è un modello per creazione di oggetti
- Una interface è un veicolo tramite cui vincolare la creazione di oggetti e istanze tramite classi
- Un type fa riferimento a una variabile, una costante, array di elementi o il valore di una key
- Un type vincola una variabile o una costante all'assegnazione di un certo tipo di valore, rendendo il codice di fatto più ordinato.

GENERICS

- Un generic type ci consente di post-porre il type ritornato da una funzione e i suoi argomenti al momento della chiamata, o il type di una property di un oggetto.
  - *Function myFunction<T>(arg: T): T { // Do something; return arg; }*
  - *doSomething<string>('Hello');*
  - *Interface book<T> { id: number, data: T }*
  - *const myBook: book<string> = { id: 1, data: '24-11-2022' }*