Presentazione TypeScript

## Che cosa è TypeScript?

TypeScript è un linguaggio di programmazione open source sviluppato da Microsoft, ideato per gli sviluppatori che sfruttano la tecnologia javascript, diinfatti è una estensione ad essa.  
Typescript è un linguaggio tipizzato, ovvero aggiunge una definizione di tipo statico , per questo è un linguaggio molto forte permettendo di avere un codice piu’ pulito e meno error-prome. Typescript usa il suo compilatore per carpire ciò che stiamo istruendo, in questo modo anche il nostro editor sarà piu’ intelligente e farà un before control degli errori di scripting, indicando in modo accurato i vari errori di esecuzione. pertanto avremo un codice anche più sicuro e ottimizzato.

il contro è che al momento TypeScript non è ancora supportato dai browser o dalle LTS di node.js.

### Il compilatore TS Perchè è necessario?

Il compilatore TS è necessario come è spiegato prima. Perchè al momento non è possibile far eseguire del codice sui browser. Pertanto abbiamo bisogno di un compilatore per convertire il nostro codice typescript in javascript, in questo modo abbiamo assoluta compatibilità nelle varie tecnologie supportate da java.

### La Type Interface

La Type Interface è un dichiarazione in modo specifico il tipo del valore delle nostre variabili, funzioni e oggetti. esistono le seguenti dichiarazione di tipo:

Primitivi:

String = ”testo”,

number= 2,

boolean= true | false,

undefined= undefined,

null= null,

any= (qualsiasi valore ma è sconsigliato perchè non avremmo piu’ le funzionalità che ci offre TypeScript)

Specificando il tipo potremmo evitare errori di scripting per es. se dichiariamo una variabile di tipo String noi non potremmo assegnare un valore di tipo numero o booleano.  
  
let Nome : String = “Mario” //OK

let Nome : String = 2 // NO, Il compilatore ci avviserà che il valore è un numero e non è un stringa per tanto lui non compilerà affinché non sarà il valore corretto richiesto.

### Il tipo ‘any’

Il tipo ‘any’ è un tipo generico che ci permette di assegnare qualsiasi tipo alla variabile che abbiamo dichiarato. Questo infatti e come se noi stiamo scrivendo semplicemente in Javascript, non ci saranno errori di compilazione la variabile sarà riassegnabile sempre anche di tipo diversi es.

let Nome : any = “Mario”;

Nome = 2;

Nome = true;

Il motivo per cui è sconsigliato utilizzarlo e che non avremmo il controllo degli errori di TypeScript ma soprattutto TypeScript non riuscirà a riconoscere che tipo di valore è la variabile che abbiamo inizializzato. Ergo non avremmo né il controllo degli codici e ne gli strumenti che ci offre TypeScript.

### Il tipo Union

Il tipo Union la sua key è ‘ | ‘ ci permette di avere una dinamicità del tipo di valore che può assumere la variabile.Il compilatore in automatico capirà in base al tipo di variabile che è stato assegnato.

es.

let Age : number | String;  
può accettare sia il valore numerico e sia il valore di carattere.

age = 20; // il compilatore saprà che è numerico pertanto è possibile di utilizzare tutte le funzionalità di metodo di quel tipo di variabile. ( .value() )

age = “20”; // il compilatore saprà che è Carattere pertanto è possibile di utilizzare tutte le funzionalità di metodo di quel tipo di variabile. ( .toLowerCase() )

### Il tipo Tuple

Il tipo Tupla serve per la dichiarazione specifica di tipo degli elementi che un array conserva,

dandoci il massimo controllo dell costruzione del array differendo gli elementi per tipo diverso e definisce anche la lunghezza del array dichiarata.

es.   
Senza Tupla :

let NomeEta : (Stringa | number ) [] or Array <String|number> = [ “Mario”, “ Luigi” , 2, ”Rossi”,4]

//Possiamo dichiarare quanti elementi vogliamo e il tipo di variabili che abbiamo dichiarato , in questo modo non avremmo il controllo dell’ nostro array poichè le nostre posizioni non hanno un tipo specifico ma può assumere anche il tipo diverso in cui lo abbiamo inizializzato.

NomeEta[0]= 2 // Possiamo assegnare apposto di “Mario” (Stringa) il valore tipo numerico.

Se vogliamo avere un controllo assoluto dovremmo utilizzare una tupla.

es.  
Con Tupla:

let NomeEta : [string, number, string] = [“Mario” , 2, “Rossi” ]

//Dichiarandolo come tupla noi avremmo massimo controllo e ottimizzazione del codice,

poiché l 'indice sarà definita dal tipo esatto e non più in modo “generico”.

pertanto non avremmo la possibilità di aggiungere o di togliere elementi dell’array avrà e dovrà avere la stessa lunghezza e tipo.

quindi noi possiamo sostituire l’elemento con un altro elemento dello stesso tipo

es.

NomeEta[0] = “Luigi”

### 

### Le Interfaces in TS

Le Interfaces in Ts ci permettono di definire la nostra struttura in modo specifico di tipo.

quindi potremmo dichiarare che la nostra variabile di oggetto è assegnare il nostro personalizzato tipo tramite L’interfaces. Come?

dovremmo innanzitutto pensare come vogliamo strutturare l’oggetto.

per esempio il nostro oggetto sarà Persona.

Persona avrà i seguenti parametri di oggetto. (nome, cognome, eta)

per far sì che TypeScript riconosca che il nostro oggetto avrà quei tipi di parametri e il tipo useremo un interface.

es.

interface Persona {

name : string,

cognome : string,

età : number

}

ok, dopo che abbiamo creato la nostra interfaccia Persona dobbiamo implementarlo alla nostra variabile oggetto Persona.

es.  
ler persona : Persona = {

name : “Mario”,

cognome : “Rossi”,

età : 30

}

in questo modo il nostro compilatore saprà esattamente come è fatto Persona, per cui i suoi attributi saranno facilmente accessibili e utilizzabili con le proprie metodologie di funzione.