Spiegazione generica (Cos'è, a cosa serve, che problema risolve, differenze con JS semplice)?

TypeScript è un linguaggio di programmazione open source sviluppato da Microsoft. Si tratta di un superset di JavaScript, il che significa che ogni programma JavaScript è anche un programma TypeScript, quindi qualunque programma scritto in JavaScript è in grado di funzionare anche con TypeScript senza apportare nessuna modifica. Lo scopo principale di TypeScript è migliorare la manutenibilità e la scalabilità del codice JavaScript attraverso l'introduzione dei tipi statici. Permette di scrivere codice più pulito. TypeScript converte automaticamente il codice in JavaScript.

Il compilatore TS (perché è necessario? e come si usa?)

Il compilatore TypeScript serve per convertire il codice TypeScript in JavaScript.
Per prima cosa si installa TypeScript via npm usando il seguente comando:

```
npm install -g typescript
```

Questo comando serve per installare TypeScript globalmente, permettendo così di usare il comando tsc su qualunque progetto futuro.

Quando si crea un file TypeScript (con un'estensione .ts) si può compilare in normale JS invocando il compilatore TypeScript con il comando:

```
tsc  <nome-del-file>
```

Questo genererà in automatico il file con estensione .js con lo stesso nome, contenente il codice compilato.

Si può anche fare in modo che i file si compilino automaticamente ad ogni salvataggio con il comando:

```
tsc  <nome-del-file> -w
```

La Type Inference

Con la Type Inference, TypeScript riesce a determinare automaticamente il tipo di una variabile grazie alla sua assegnazione iniziale. Ciò consente agli sviluppatori di sfruttare i vantaggi dei tipi statici senza doverli dichiarare in ogni parte del codice.

Il tipo ‘any’

Il tipo 'any' è utilizzato per rappresentare una variabile senza specificarne un tipo. Quindi la variabile potrà contenere qualsiasi tipo di dato. Il tipo any può essere applicato anche agli array e ad un oggetto ed invocare un metodo

Il tipo Union

Union consente di specificare che una variabile può avere uno tra diversi tipi. Ad esempio: string | number indica che la variabile può essere una stringa o un numero.

Il tipo Tuple

Il tipo Tuple permette di definire un array con un numero fisso di elementi, e di definire array con elementi di tipi diversi tra loro quindi ognuno con un tipo specifico.
Ad esempio:

```
let persona: [string, number] = ["Mario Rossi", 25];
```

Dove il primo elemento è di tipo stringa essendo il nome, mentre il secondo essendo età sarà di tipo numerico

Le Interfaces in TS

Le interfacce in Typescript sono in grado di specificare nel dettaglio la struttura di un oggetto: quante e quali proprietà/metodi debba avere. Le interfacce possono essere anche definite partendo dalla definizione di altre interfacce, estendendo i loro contenuti ed ereditando le coppie chiave-tipo già definite. Il nome delle interfacce viene definito tramite PascalCase.

Types vs Interfaces

Le interfacce sono più adatte per dichiarare contratti, mentre i types sono più flessibili e possono rappresentare tipi di dati arbitrari.

I Generics

I generics consentono di scrivere codice in modo flessibile, essi permettono di definire funzioni, classi o interfacce che sono in grado di lavorare con diversi tipi di dati senza specificarli esplicitamente. Per esempio, è possibile creare una funzione in cui il tipo dei parametri e del valore di ritorno non viene definito inizialmente ma va fornito al momento in cui la funzione viene invocata. Questo permette anche la riutilizzabilità del codice