CLASSI ASTRATTE

In TypeScript, una classe astratta è una classe che non può essere istanziata direttamente, ma può essere utilizzata solo come classe base per altre classi derivate. Una classe astratta definisce un'interfaccia comune e una serie di metodi che le classi derivate devono implementare.

Le classi astratte sono utili quando si desidera definire un comportamento di base che deve essere condiviso tra diverse classi, ma non ha senso creare un'istanza diretta della classe astratta stessa. Le classi derivate possono estendere la classe astratta e implementare i metodi astratti, fornendo così un'implementazione specifica del comportamento definito nella classe astratta.

typescript Copy code

```typescript
abstract class NomeClasseAstratta {
  // Proprietà e metodi
}
```

Per definire un metodo astratto, si utilizza la parola chiave abstract prima della dichiarazione del metodo. Un metodo astratto non ha un'implementazione nella classe astratta e deve essere implementato nelle classi derivate.

```typescript
abstract class Veicolo {
  protected velocita: number;

  constructor(velocita: number) {
    this.velocita = velocita;
  }

  abstract accelera(): void;

  frena(): void {
    console.log("Il veicolo si ferma.");
    this.velocita = 0;
  }
}

class Automobile extends Veicolo {
  accelera(): void {
    console.log("L'automobile accelera.");
    this.velocita += 10;
  }
}

class Moto extends Veicolo {
  accelera(): void {
    console.log("La moto accelera.");
    this.velocita += 5;
  }
}

// Creazione di istanze delle classi derivate
const automobile = new Automobile(0);
const moto = new Moto(0);

automobile.accelera(); // Output: L'automobile accelera.
console.log(automobile.velocita); // Output: 10

moto.accelera(); // Output: La moto accelera.
console.log(moto.velocita); // Output: 5

automobile.frena(); // Output: Il veicolo si ferma.
console.log(automobile.velocita); // Output: 0
```

Le classi astratte in TypeScript consentono quindi di definire una struttura gerarchica comune per le classi derivate, garantendo che abbiano determinati metodi definiti e consentendo allo stesso tempo di personalizzare l'implementazione specifica per ciascuna classe derivata.

Alcune situazioni in cui può essere necessario utilizzare una classe astratta:

Definizione di una base comune: Se hai un insieme di classi correlate che condividono una logica comune o un insieme di metodi simili, puoi utilizzare una classe astratta per definire questa logica comune. Le classi derivate possono estendere la classe astratta e implementare o estendere i metodi specifici. In questo modo, puoi evitare la duplicazione del codice e garantire una struttura coerente tra le classi derivate.

Contratto di implementazione: Puoi utilizzare una classe astratta per definire un contratto che le classi derivate devono rispettare. La classe astratta può dichiarare metodi astratti, che devono essere implementati dalle classi derivate. Questo garantisce che tutte le classi derivate forniscono l'implementazione corretta dei metodi richiesti. Le classi astratte possono anche fornire un'implementazione di base per alcuni metodi, che possono essere sovrascritti dalle classi derivate se necessario.

Polimorfismo: Le classi astratte consentono di utilizzare il concetto di polimorfismo, che significa che un oggetto di una classe derivata può essere trattato come un'istanza della classe astratta. Questo consente di scrivere codice più generico ed estendibile, in cui le operazioni comuni possono essere eseguite su oggetti di diverse classi derivate attraverso il riferimento alla classe astratta.

Incapsulamento: Utilizzando una classe astratta come punto di accesso principale per un insieme di classi derivate, puoi nascondere i dettagli di implementazione delle classi derivate e fornire un'interfaccia più generica. Ciò aiuta a mantenere il codice più modulare e facilita l'estensibilità e la manutenzione del sistema.