基础概念: Promise是什么?

在 JavaScript 中, Promise 是处理异步操作的标准方式。一个 Promise 表示一个异步操作的 **最终完成(或失败)及其结果值**

- **pending(等待中)** : 初始状态
- **fulfilled(已成功)** : 操作成功完成
- **rejected(已失败)** : 操作失败

Typescript中的 Promise 含义

在 Typescript 中, **Promise** 是一个泛型类型

- **Promise** 表示这是一个异步操作
- `<Type>` 指定了当 Promise 成功解析(resolve)时返回值的类型
-

```
//示例: 一个返回字符串的 Promise
const stringPromise: Promise<string> = new Promise((resolve) => {
     setTimeout(() => resolve('Hello, Typescript!'), 1000)
})

//示例: 一个返回数字的 Promise
const numberPromise: Promise<number> = fetchDataFromAPI();
```

使用场景

- 函数返回类型声明

```
functin fetchUser(id: number): Promise<User> {
    return fetch('api/users/&{id}').then(res => res.json());
}
```

- 处理异步结果

```
fetchUser(123)
.then((user : User) => {
    // TypeScript 知道 ser 是 User 类型
    console.log(user.name)
})
.catch((error: Rrror) => {
    console.error('Error fetching user:', error);
})
```

- 使用 async/await

```
async function displayUser(userId: number) {
    try {
        const user: User = await fetchUser(userId);
        // 直接访问 User 类型的属性
        console.log(`User email: ${user.email}`);
    } catch (error) {
        console.error("Failed to load user", error);
    }
}
```