Контракты в TypeScript — это определения типов, которые указывают структуру и требования к определённому объекту, например объекту или функции. Они определяют форму и поведение объектов и функций, обеспечивая безопасность типов и согласованность

Вот несколько ключевых моментов о контрактах в TypeScript:

1. Интерфейсы: Интерфейсы используются для определения структуры объекта, включая свойства и методы. Например:

interface User {

id: number;

name: string;

email: string;

}

2. Типы: Вы также можете использовать типы для создания сложных структур данных. Например:

type Point = {

x: number;

y: number;

};

3. Дженерики: TypeScript поддерживает дженерики, которые позволяют создавать контракты, которые работают с различными типами данных. Например:

function identity<T>(arg: T): T {

return arg;

}

4. Типы функций: Вы можете описывать контракты для функций, определяя их параметры и возвращаемые значения:

type Callback = (data: string) => void;

function fetchData(callback: Callback) {

callback("Данные загружены");

}

5. Наследование интерфейсов: Интерфейсы могут наследоваться друг от друга, что позволяет создавать более сложные структуры.

interface Employee extends User {

position: string;

}

Использование контрактов в TypeScript помогает поддерживать чистоту и предсказуемость кода, облегчая его понимание и сопровождение.

**Некоторые преимущества использования контрактов в TypeScript:**

**Безопасность типов.**

Контракты помогают ловить потенциальные ошибки типов во время компиляции, уменьшая вероятность ошибок во время выполнения.

**Согласованность кода.**

Принуждение к соблюдению контрактов гарантирует, что все компоненты придерживаются одинаковых стандартов и интерфейсов.

**Документация.**

Контракты служат документацией для кода, облегчая разработчикам понимание и работу с кодовой базой.

**Рефакторинг.**

Контракты облегчают рефакторинг кода, предоставляя чёткое руководство, как должны взаимодействовать разные части приложения.