**Перечисления (enums)**— это специальный тип, который позволяет разработчикам определять набор именованных констант. Это делает код более читаемым и управляемым, особенно когда нужно работать с фиксированным набором значений.

**Типы перечислений**

TypeScript поддерживает два основных типа перечислений:

**1. Числовые перечисления:**

По умолчанию, члены перечисления получают числовые значения, начиная с 0 и увеличиваясь на единицу для каждого следующего члена.

Если установить для North = 1, то перечисления идут от 1

enum Direction {

North,

East,

South,

West

}

console.log(Direction.North); // 0

console.log(Direction.East); // 1

console.log(Direction[1]); // East

**2. Строковые перечисления:**

Каждый член перечисления имеет строковое значение, что делает их более понятными и удобными для чтения.

enum Direction {

North = "North",

East = "East",

South = "South",

West = "West"

}

console.log(Direction.North); // "North"

|  |  |  |
| --- | --- | --- |
| Критерий | enum | Объект |
| Типобезопасность | Поддерживает типобезопасность в TypeScript | Не обеспечивает строгую типизацию (необходима задавать as const) |
| Инициализация | Члены перечисления могут быть автоматически инициализированы | Значения должны быть явно заданы |
| Использование | Можно использовать в качестве типа для переменных и параметров функций  function Direction (direction: Direction) | Может использоваться как обычный объект с произвольными свойствами  function Direction (direction: typeof Direction [keyof typeof Direction]) |
| Существование на этапе выполнения | В зависимости от типа (обычный или `const`) может не существовать в скомпилированном коде | Всегда существует как объект в JavaScript |

Вот несколько случаев, когда целесообразно использовать enum:

1. Представление фиксированных наборов значений: Если у вас есть набор связанных значений, таких как дни недели, месяцы, состояния (например, "активный", "неактивный"), использование enum позволяет сгруппировать их.

2. Улучшение читаемости кода\*: Вместо использования магических чисел или строк, вы можете использовать имена из enum, что делает код более понятным. Например, вместо `status === 1` можно написать `status === Status.Active`.

3. Типизация: enum предоставляет строгую типизацию, что позволяет избежать ошибок, связанных с использованием неправильных значений.

4. Легкость поддержки: Если в будущем вам нужно изменить или добавить новое значение, вы можете сделать это в одном месте (в самом `enum`), а не искать по всему коду.

5. Использование в switch-операторах: Перечисления хорошо работают с конструкцией `switch`, что позволяет удобно обрабатывать различные состояния.

**Заключение**

Перечисления (enum) в TypeScript предоставляют мощный инструмент для работы с фиксированными наборами значений, обеспечивая при этом типобезопасность и читаемость кода. В то время как объекты также могут использоваться для хранения связанных данных, перечисления предлагают дополнительные преимущества в контексте строгой типизации и удобства использования в разработке.