ТИП UNKNOWN

Бывают ситуации, когда мы можем получить сущность неизвестного типа. Для работы с ними существует **тип unknown**, который является **типобезопасным аналогом any:**

```ts
let smth: any;

smth = 'str';

let data: string[] = smth;
data.find(e => e);
```

При использовании **any** мы не получим ошибку в 6й строке. Но в рантайме она будет

```ts
let smth: unknown;

smth = 'str';

let data: string[] = smth;
data.find(e => e);
```

При использовании **unknown** ошибка будет. Нельзя применять метод неизвестно к чему

Опасность any в том, что это **любой тип**. В нем нет никакой строгости. Никакие проверки типов в нем не выполняются. А вот unknow – **это неизвестный тип.**
В any может быть **что угодно**, а в unknown – мы **не знаем** что может быть. К чему угодно может применятся **что угодно**, а к неизвестному – **ничего**

```ts
const someValue: any = 10;
someValue.method(); // Ok
```

```ts
const someValue: unknown = 10;
someValue.method(); // Error
```

Для работы с этим типом необходимо использовать **сужение типов**. Так мы поймем что это и сможем правильно с ним работать:

```ts
function fetchData(data: unknown): void {
    if (typeof data === "string") {
        console.log(data.toLowerCase());
    }
    data  // тут unknown
}
```

ПРИМЕНЕНИЕ UNKNOWN

Данный тип можно использовать для работы с функциями, которые возвращают что-угодно. Например, JSON.parse(). Так мы избежим ошибок и правильно будем работать с данными:

```typescript
const userData =
    '{"isBirthdayData": true, "ageData": 40, "userNameData": "John"}';

function safeParse(s: string): unknown {
    return JSON.parse(s);
}

const data = safeParse(userData);

function transferData(d: unknown): void {
    if (typeof d === "string") {
        console.log(d.toLowerCase());
    } else if (typeof d === "object" && d) {
        console.log(data);
    } else {
        console.error("Some error");
    }
}

transferData(data);
```

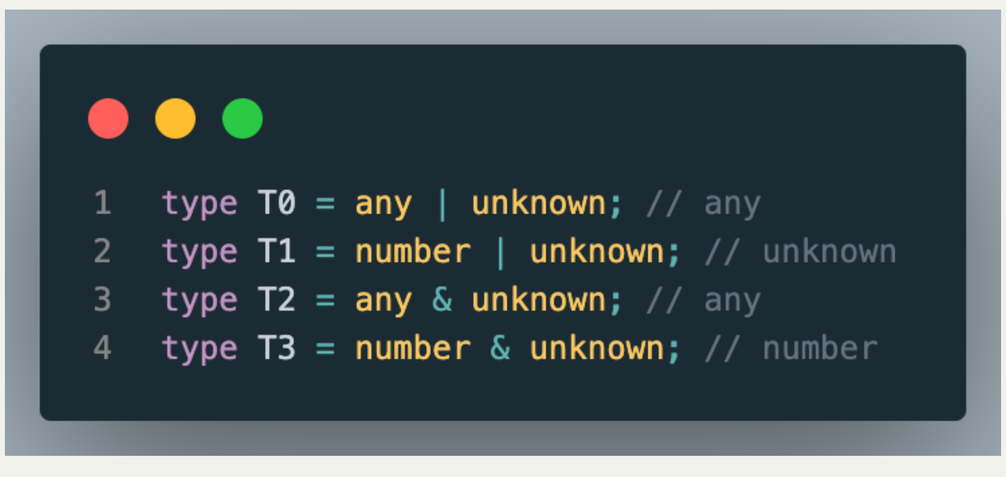
В JSON строке могут быть разные данные и благодаря функции safeParse мы получим не что угодно, а неизвестно что. А функция transferData использует сужение типов для правильной работы

В TS, в конструкции **try/catch** ошибка, приходящая в catch будет типа **unknown**. Это происходит из-за того, что компилятор не знает, какую именно ошибку разработчик “выкинет” из блока try. Это может быть строка, экземпляр определенного класса или что-то еще. Так что здесь тоже стоит применять сужение типов:

```typescript
try {
    if (1) {
        throw new Error("error");
    }
} catch (e) {
    if (e instanceof Error) {
        console.log(e.message);
    } else if (typeof e === "string") {
        console.log(e);
    }
}
```

Работа с union и intersection типами:

- Если тип unknown составляет тип объединение (union), то он перекроет **все типы**, за исключением типа any
- Если тип unknown составляет тип пересечение (intersection), то он **будет перекрыт всеми типами**



```typescript
type T0 = any | unknown; // any
type T1 = number | unknown; // unknown
type T2 = any & unknown; // any
type T3 = number & unknown; // number
```