В TypeScript тайпгарды (type guards) — это функции, которые позволяют проверить, принадлежит ли значение определённому типу. Они используются для сужения типов данных (type narrowing) внутри программы. Тайпгарды реализуются как специальные предикаты, которые возвращают true или false и при этом уточняют тип данных.

**1. Проверка примитивных типов**

Примитивные типы, такие как string, number, boolean, можно проверять с **помощью оператора typeof.**

**String**

function isString(value: unknown): value is string {

return typeof value === 'string';

}

**Number**

function isNumber(value: unknown): value is number {

return typeof value === 'number';

}

**Boolean**

function isBoolean(value: unknown): value is boolean {

return typeof value === 'boolean';

}

**2. Проверка массивов**

Для проверки массивов можно использовать метод Array.isArray().

function isArray(value: unknown): value is Array {

return Array.isArray(value);

}

**3. Проверка объектов**

Проверка объектов выполняется через проверку на null (объект в TypeScript может быть null) и уточнение свойств.

function isObject(value: unknown): value is object {

return typeof value === 'object' && value !== null;

}

**4. Проверка пользовательских (кастомных) типов**

Для проверки пользовательских типов удобно использовать типы данных с фиксированными свойствами.

Пример с интерфейсом

interface User {

name: string;

age: number;

}

function isUser(value: unknown): value is User {

return (

typeof value === 'object' &&

value !== null &&

'name' in value &&

'age' in value &&

typeof (value as User).name === 'string' &&

typeof (value as User).age === 'number'

);

}

**5. Проверка объединённых типов (Union Types)**

TypeScript позволяет использовать тайпгарды для проверки значений в объединённых типах.

**Пример с string | number**

function isStringOrNumber(value: unknown): value is string | number {

return typeof value === 'string' || typeof value === 'number';

}

if (isStringOrNumber(value)) {

if (typeof value === 'string') {

console.log(value.toUpperCase()); // Уточнение до string

} else {

console.log(value.toFixed(2)); // Уточнение до number

}

}

**6. Проверка через instanceof**

Для проверки объектов, созданных через конструкторы (классы), используется instanceof.

class MyClass {

constructor(public name: string) {}

}

function isMyClass(value: unknown): value is MyClass {

return value instanceof MyClass;

}

console.log(value.name); // value считается MyClass

}

**7. Проверка HTML элементов**

function isHTMLInputElement(element: HTMLElement): element is HTMLInputElement {

return element.tagName === 'INPUT';

или

return element instanceof HTMLInputElement

}

function isHTMLDivElement(element: HTMLElement): element is HTMLDivElement {

return element.tagName === 'DIV';

}

Где это можно использовать?

function handleHTMLElement(element: HTMLElement) {

if (isHTMLInputElement(element)) {

element.value = "Hello!"; // Теперь TypeScript знает, что это HTMLInputElement

} else if (isHTMLDivElement(element)) {

element.innerText = "Hello!"; // TypeScript понимает, что это HTMLDivElement

} else {

console.log("Это какой-то другой элемент:", element);

}

}

**Тайпгард с обработкой событий (Event Target)**

function isHTMLElement(target: EventTarget | null): target is HTMLElement {

return target instanceof HTMLElement;

}

function isHTMLButtonElement(target: EventTarget | null): target is HTMLButtonElement {

return target instanceof HTMLButtonElement;

}

Пример использования:

document.addEventListener("click", (event) => {

const target = event.target;

if (isHTMLElement(target)) {

console.log("Это HTML-элемент:", target.tagName);

if (isHTMLButtonElement(target)) {

console.log("Это кнопка!", target.innerText);

}

}

});