РАБОТА С DOM В TS

Особенностью работы с DOM-структурой в TS является то, что он предоставляет множество специальных интерфейсов для этого. Существует интерфейс **документа** (*Document*), **различных событий** (*Event, MouseEvent и тп*) и **элементов на странице**. Последние подчиняются строгой иерархии в зависимости от специфичности:

Node ↓	Любой узел на странице: текстовый, содержимое изображения, любые элементы... Содержит только самые базовые свойства и методы
Element ↓	Любые элементы на странице, в том числе и невидимые на странице. Содержит базовые свойства и методы, присущие всем элементам. Получаем его без утверждения типа
HTMLElement ↓	Любые html-элементы на странице. Содержит более специфичные свойства и методы, присущие элементам
HTMLInputElement HTMLParagraphElement HTMLAnchorElement и другие	Определенные html-элементы на странице. Содержат специфичные для них свойства и методы в дополнение к общим

Все эти интерфейсы нужны для четкого указания с чем мы работаем и правильного доступа к нужным **свойствам/методам**

```ts
const box = document.querySelector(".box") as HTMLElement;
const p = document.querySelector(".paragraph") as HTMLParagraphElement;
const input = document.querySelector("a"); // Автоматическое определение
```

Если мы указываем селектор, по которому **и так понятно, что за элемент будет получен** - TS автоматически подставит нужный тип интерфейса. Утверждение в таком случае **не нужно**
Такая же ситуация и при создании новых элементов через команду **createElement()**

Помните, утверждение типа избавляет вас от null только в TS. Всегда есть шанс не получить элемент на странице. Так что не забывайте про стандартные средства защиты (try/catch, if и тд.)