08

Sobre casting explícito

Em TypeScript, podemos referenciar um dado de um tipo mais especializado através de um tipo mais genérico. Por exemplo:

```
let x: Element;
let y: HTMLInputElement;
x = y; // funciona!
```

O código acima é possível, porque todo `HTMLInputElement` é um `Element`. O que ocorre é um casting implícito, no qual o desenvolvedor não precisa atuar. Contudo, nem todo Element é um HTMLInputElement.

Por isso não podemos fazer:

```
let x: Element;
let y: HTMLInputElement;
y = x; // erro
```

Temos o erro:

```
Type 'Element' is not assignable to type 'HTMLInputElement'.
  Property 'accept' is missing in type 'Element'.
```

Contudo, quando usarmos `document.querySelector()` o TypeScript considera que o retorno será sempre do tipo `Element`. Justo, pois `document.querySelector` pode retornar um HTMLInputElement, HTMLTableElement, HTMLAnchorElement, etc. O que todos eles possuem em comum, são elementos do DOM, por isso podem ser tratados como `Element`.

Contudo, `Element` não expõe propriedades e métodos de cada um dos tipos específicos que listamos no parágrafo anterior e isso pode nos causar problemas em nosso código, por exemplo, para pegar o `value` de um `HTMLInputElement`.

Aprendemos que podemos realizar um casting **explícito** de um tipo mais genérico para um tipo mais específico.

Marque a opção abaixo que realiza corretamente um casting explícito:

```
let tabela = <HTMLTableElement> document.querySelector('table');
```

Correto! Realizamos o casting explícito de `Element` para `HTMLTableElement`. Inclusive, devido ao casting, o TypeScript infere que o tipo de `tabela` será `HTMLTableElement`.

B

```
let tabela: HTMLTableElement = document.querySelector('table');
```

C

```
let tabela = (HTMLTableElement) document.querySelector('table');
```

PRÓXIMA ATIVIDADE