Operador de Tipo Typeof

O operador de tipo `typeof`

O JavaScript já possui um operador `typeof` que você pode usar em um contexto de expressão:

```
// Imprime "string"
console.log(typeof "Hello world");
```

O TypeScript adiciona um operador `typeof` que você pode usar em um contexto de tipo para se referir ao tipo de uma variável ou propriedade:

```
let s = "hello";
let n: typeof s;

let n: string
```

Isso não é muito útil para tipos básicos, mas combinado com outros operadores de tipo, você pode usar `typeof` para expressar muitos padrões. Por exemplo, vamos olhar para o tipo pré-definido `ReturnType<T>`. Ele recebe um tipo de função e produz seu tipo de retorno:

```
type Predicate = (x: unknown) => boolean;
type K = ReturnType<Predicate>;

type K = boolean
```

Se tentarmos usar `ReturnType` em um nome de função, vemos um erro instrutivo:

```
function f() {
  return { x: 10, y: 3 };
}
type P = ReturnType<f>;
'f' refere-se a um valor, mas está sendo usado como tipo aqui. Você quis dizer
'typeof f'?
```

Lembre-se de que valores e tipos não são a mesma coisa. Para se referir ao tipo que o valor `f` possui, usamos `typeof`:

```
function f() {
  return { x: 10, y: 3 };
}
type P = ReturnType<typeof f>;

type P = {
    x: number;
    y: number;
}
```

Limitações

O TypeScript limita intencionalmente os tipos de expressões que você pode usar com `typeof`.

Especificamente, é legal usar `typeof` apenas em identificadores (ou seja, nomes de variáveis) ou suas propriedades. Isso ajuda a evitar a confusão de escrever código que você acha que está executando, mas não está:

```
// Destinado a usar = ReturnType<typeof msgbox>
let shouldContinue: typeof msgbox("Você tem certeza de que deseja continuar?");
',' esperado.
```