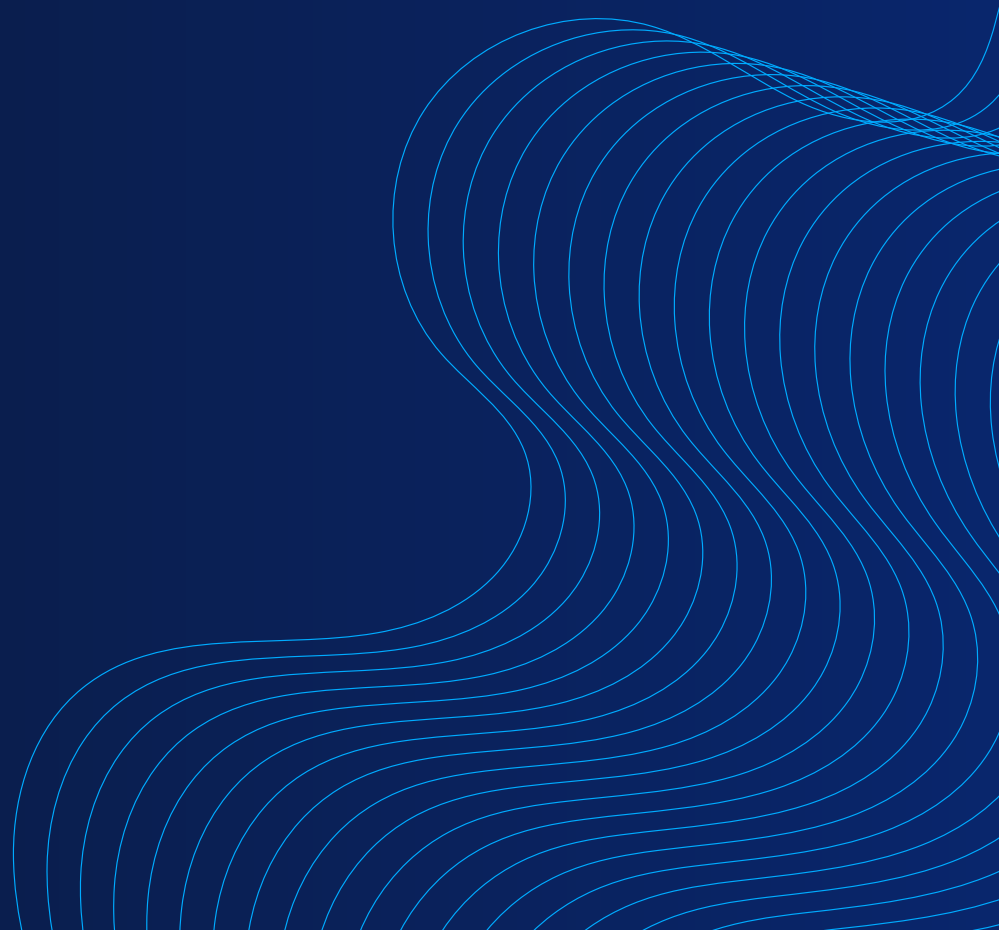


# O que são Types e Union Types



Types ou Tipos são a base do sistema de tipos do TypeScript. Eles são usados para especificar a forma que as variáveis, funções, ou propriedades devem ter.

Os tipos podem ser primitivos, como `number`, `string`, e `boolean`, ou complexos, como arrays, objetos, ou tipos personalizados.

## Tipos Primitivos: Representam valores simples.

- **string**: Representa uma cadeia de caracteres.
- **number**: Representa valores numéricos, incluindo inteiros e decimais.
- **boolean**: Representa valores booleanos, que podem ser `true` ou `false`.




```
type Nome = string;  
type Idade = number;  
  
let meuNome: Nome = "Maria";  
let minhaldade: Idade = 25;
```

Aqui, `nome`, `idade`, e `ativo` são variáveis associadas a tipos primitivos (`string`, `number`, `boolean`), o que significa que elas só podem armazenar valores compatíveis com esses tipos.

## Tipos Complexos

- Arrays: Um tipo de coleção de elementos. No TypeScript, os arrays são tipados, como `number[]` para um array de números.
- Objetos: Estruturas que contêm um conjunto de pares chave-valor, onde cada valor pode ter seu próprio tipo.
- Tuplas: Arrays com um número fixo de elementos, onde cada elemento pode ter um tipo diferente.



```
type Endereco = {  
  rua: string;  
  numero: number;  
  cidade: string;  
  estado: string;  
};  
  
let enderecoResidencial: Endereco = {  
  rua: "Rua das Flores",  
  numero: 123,  
  cidade: "São Paulo",  
  estado: "SP"  
};
```

Aqui, o tipo `Endereco` é uma estrutura que combina várias propriedades (`rua`, `numero`, `cidade`, `estado`), cada uma com seu respectivo tipo. Isso é útil para modelar dados complexos de forma clara e coesa.

# Union Types (Tipos de União)

Union Types permitem que uma variável ou parâmetro aceite mais de um tipo.

Isso é útil em situações onde um valor pode ser de diferentes tipos.

Para definir um Union Type, utiliza-se o operador `|` entre os tipos possíveis.

```
type Identificador = number | string;
```



```
let id: Identificador;  
id = 101; // Válido  
id = "202A"; // Também válido
```

O tipo `Identificador` pode ser tanto um `number` quanto uma `string`. Esse tipo de declaração é particularmente útil em APIs ou funções que precisam lidar com diferentes formatos de entrada.

# Vantagens de Usar `type` para Criar Tipos Personalizados

**Legibilidade:** Tipos personalizados tornam o código mais fácil de entender, fornecendo contexto sobre o que uma variável ou função representa.

**Reusabilidade:** Uma vez criado, um tipo personalizado pode ser reutilizado em várias partes do código, promovendo consistência.

**Manutenibilidade:** Alterações nos tipos personalizados afetam automaticamente todas as variáveis e funções que utilizam esses tipos, facilitando a manutenção.

**Robustez:** Com tipos definidos explicitamente, o TypeScript pode detectar e alertar sobre inconsistências e erros em tempo de compilação, antes que o código seja executado.

# Conclusão

---

As declarações de tipos em TypeScript, especialmente com o uso da palavra-chave **'type'**, são uma poderosa ferramenta para criar código mais seguro, legível e fácil de manter.

Ao definir tipos personalizados, os desenvolvedores conseguem capturar as nuances do domínio da aplicação, assegurando que o código seja robusto e consistente em toda a base de código.





# E aí, curtiu?

Esperamos que esse resumo tenha enriquecido sua perspectiva estratégica para enfrentar os desafios.

Salve esse PDF para consultar sempre que precisar.