Propriedades Opcionais no typescript

No TypeScript, é possível definir propriedades opcionais em um tipo ou interface usando o operador `?` após o nome da propriedade. Isso indica que a propriedade pode estar presente no objeto, mas não é obrigatória.

Por exemplo, suponha que você queira definir um tipo para um objeto que representa uma pessoa, mas a propriedade "email" pode ou não estar presente. Você pode definir o tipo `Pessoa` da seguinte maneira:

typescript Copy code

```typescript
type Pessoa = { nome: string; idade: number; email?: string; };
```

Observe que a propriedade `email` é seguida pelo operador `?`, o que a torna opcional.

Com isso, você pode criar um objeto com o tipo `Pessoa` sem a propriedade `email` ou com ela:

typescript Copy code

```typescript
const pessoa1: Pessoa = { nome: "João", idade: 30, }; const pessoa2: Pessoa = { nome: "Maria", idade: 25, email: "maria@example.com", };
```

Ao fazer isso, a propriedade `email` pode ser `string` ou `undefined`, pois essa propriedade é opcional.

Ao definir uma propriedade como opcional, você pode acessá-la usando a sintaxe de ponto `.` normalmente, mas deve estar ciente de que ela pode estar `undefined` em alguns casos. Por exemplo:

typescript Copy code

```typescript
if (pessoa1.email) { console.log(pessoa1.email.toUpperCase()); // Erro: pessoa1.email pode ser undefined } if (pessoa2.email) { console.log(pessoa2.email.toUpperCase()); // OK }
```

Observe que, ao tentar acessar `pessoa1.email.toUpperCase()`, um erro é gerado porque `email` pode ser `undefined` e não tem o método `toUpperCase()`. Portanto, é necessário verificar se a propriedade existe antes de acessá-la.
