React

Prof. José Matheus

Convertendo um projeto JS para TS

Abaixo o passo a passo para transformar um projeto Javascript para Typescript:

1. Crie um novo projeto React com o Vite, utilizando o comando `npm init vite`, selecione Typescript;
2. Instale as dependências com o comando `npm install react-router-dom axios --save`;
3. Instale o typescript e os tipos necessários `npm install -D typescript @types/react @types/react-dom`;
4. Copie para o novo projeto todos os arquivos da pasta src;
5. Renomeie todos os arquivos com a extensão .jsx para .tsx e .js para .ts, exceto o index.js, que vai para .tsx também;
6. Modifique os arquivos conforme necessário, importando tipagens e criando interfaces.

JS para TS

Ao utilizar JSX.Element, estamos especificando explicitamente que a função retorna um elemento JSX. Isso ajuda o TypeScript a entender melhor o código e a realizar verificações de tipo mais precisas durante a compilação. Não se trata de uma gambiarra ou solução paliativa.

Se não especificarmos o tipo de retorno ou se usarmos any como tipo de retorno, estaremos perdendo a oportunidade de aproveitar os benefícios do TypeScript, como a detecção de erros de tipo em tempo de compilação e o autocompletamento inteligente em editores de código compatíveis.

Definir a interface dentro do mesmo arquivo ou separado

Definir a interface dentro do mesmo arquivo:

Prós:

- **Conveniência:** É conveniente definir a interface no mesmo arquivo em que ela é usada, especialmente para interfaces pequenas ou específicas de um componente;
- **Visibilidade:** A interface está imediatamente visível para qualquer pessoa que esteja trabalhando no componente, facilitando a compreensão do que o componente espera como entrada ou produz como saída.

Contras:

- **Reutilização Limitada:** Se outras partes da aplicação precisarem da mesma interface, elas terão que redefinir a interface novamente, resultando em duplicação de código;
- **Poluição do Espaço Global:** Muitas interfaces definidas em muitos arquivos podem poluir o espaço global do arquivo e dificultar a manutenção.

Definir a interface dentro do mesmo arquivo ou separado

Definir a interface em um arquivo separado:

Prós:

- **Reutilização:** As interfaces podem ser facilmente reutilizadas em toda a aplicação, pois estão centralizadas em um único local;
- **Organização:** Manter interfaces em arquivos separados pode ajudar a organizar o código e facilitar a localização de interfaces específicas;
- **Prevenção de Duplicação:** Evita a duplicação de código, já que várias partes da aplicação podem usar a mesma interface.

Contras:

- **Complexidade Adicional:** Pode adicionar alguma complexidade adicional à estrutura do projeto, especialmente se houver muitas interfaces;
- **Menor Visibilidade:** As interfaces podem não estar imediatamente visíveis para quem está trabalhando em um componente específico, o que pode exigir navegação para outros arquivos para entender as expectativas de entrada/saída.

Nomenclatura de interfaces

- **Nomeclatura Descritiva:** É recomendado usar nomes descritivos que indiquem claramente a finalidade ou o papel da interface no código. Isso pode ajudar a melhorar a legibilidade e a compreensão do código;
- **Prefixos ou Sufixos:** Alguns desenvolvedores preferem usar prefixos ou sufixos para identificar interfaces, como I no início (ex: IPost) ou Interface no final (ex: PostInterface). No entanto, isso é mais uma questão de preferência pessoal ou de convenções estabelecidas em equipes ou projetos específicos;
- **Contexto e Domínio:** Levar em consideração o contexto e o domínio do problema ao nomear interfaces pode ser útil. Por exemplo, se estiver trabalhando em uma aplicação financeira, pode fazer sentido usar termos relacionados a finanças para nomear interfaces relevantes;
- **Convenções da Comunidade:** Em alguns casos, seguir as convenções de nomenclatura estabelecidas pela comunidade TypeScript pode ser útil para garantir consistência e facilidade de compreensão para outros desenvolvedores que possam trabalhar no mesmo código no futuro.

Sintaxe de importação

Quando você importa algo usando a sintaxe de desestruturação com chaves `{ }`, você está importando apenas os membros especificados do módulo. Se você estiver importando algo que é exportado como um tipo TypeScript, como uma interface ou um tipo de dados, então sim, você está importando também o tipo.

TYPESCRIPT

```typescript
// types.ts
export interface User {
  name: string
  age: number
}
```

TYPESCRIPT

```typescript
// Outro arquivo
import { User } from './types'

// Agora você pode usar o tipo
User neste arquivo
const user: User = {
  name: 'Alice',
  age: 30
}
```

Arquivos finais

Dúvidas?

Referências

React. Disponível em: <https://react.dev/>

Typescript. Disponível em: <https://www.typescriptlang.org/>