THAMIRIS ADRIANO

PDS TECH

FULL STACK DEVELOPMENT

TYPESCRIPT

AULA 01

SUMÁRIO

O QUE VEM POR AÍ?

Nessa aula, as pessoas docentes apresentarão conceitos de desenvolvimento básico de TypeScript. Você aprenderá os fundamentos, incluindo a diferença entre TypeScript e JavaScript, compilação de código TypeScript para JavaScript executável, declaração de variáveis com tipos explícitos e implícitos, bem como os tipos básicos, como números, strings e booleans. Este conhecimento é essencial para construir aplicativos web e móveis mais robustos e compreender os princípios fundamentais do TypeScript.

HANDS ON

Na primeira videoaula, faremos uma introdução ao TypeScript, um superset de JavaScript desenvolvido pela Microsoft. Os(as) estudantes aprenderão as vantagens de usar o TypeScript, como a tipagem estática, que ajuda a evitar erros de programação desde o início do desenvolvimento. Além disso, entenderão o conceito de tipagem em programação e como o TypeScript funciona.

Já na segunda videoaula, os alunos e as alunas serão introduzidos(as) a um hands on no desenvolvimento de projetos com TypeScript. Por meio de um exemplo prático, eles(as) aprenderão a configurar o ambiente de desenvolvimento, criar arquivos, inserir código TypeScript, entender o processo de transpilação para JavaScript e executar o código.

SAIBA MAIS

TypeScript é um superset do JavaScript, que adiciona recursos de tipagem estática ao ambiente de desenvolvimento web. Ele foi desenvolvido pela Microsoft e é amplamente utilizado na construção de aplicativos web e mobile modernos. A principal diferença entre TypeScript e JavaScript é a adição de tipos estáticos, que permitem aos desenvolvedores declarar o tipo de variável que estão usando. Aqui estão algumas características do TypeScript.

1. Tipagem estática: uma das características do TypeScript é a capacidade de definir tipos de variáveis, parâmetros de função e retornos de função de forma estática. Isso ajuda a evitar erros de tipo durante o desenvolvimento, tornando o código mais seguro e compreensível.
2. Melhor ferramenta de desenvolvimento: o TypeScript é altamente integrado com ferramentas populares de desenvolvimento, como o Visual Studio Code. Ele fornece recursos de autocompletar, detecção de erros em tempo real e sugestões de código, tornando o processo de desenvolvimento mais eficiente.
3. Maior escalabilidade: à medida que os projetos crescem, o TypeScript se torna uma escolha natural devido à sua tipagem estática. Isso torna mais fácil manter e estender o código conforme a base de código cresce, minimizando os erros de tipo que podem ocorrer em projetos grandes e complexos.
4. Facilidade de refatoração: com o TypeScript, é mais fácil fazer refatorações em grande escala no código, como renomear variáveis ou extrair partes do código para funções separadas, porque ele pode identificar todos os lugares em que uma variável ou função é usada.
5. Integração com ecossistema JavaScript: TypeScript é compatível com o JavaScript e pode aproveitar todas as bibliotecas e frameworks JavaScript populares. Isso permite que as pessoas desenvolvedoras usem TypeScript em projetos existentes ou em novos projetos que precisam tirar proveito de código JavaScript pré-existente.

Após escrever o código em TypeScript, é necessário transformá-lo em código JavaScript, pois os navegadores não conseguem entender TypeScript diretamente. O código JavaScript gerado pode ser executado em qualquer ambiente que suporte JavaScript, tornando-o amplamente compatível e utilizável.

O TypeScript fornece um compilador que faz essa conversão. Isso é feito executando o comando tsc (TypeScript Compiler) no terminal.

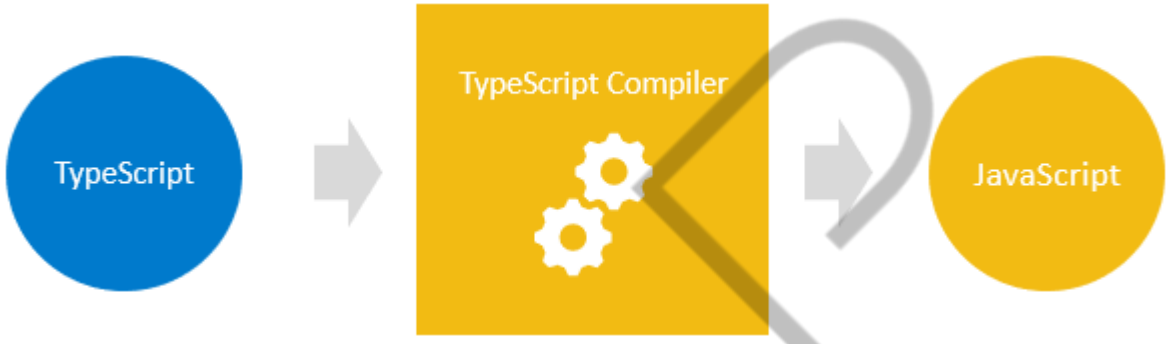

Figura 1 – Processo de compilação
Fonte: Typescript Tutorial (2020)

A compilação é uma etapa crucial no TypeScript, pois transforma o código TypeScript em JavaScript, tornando-o executável em ambientes como navegadores da web e servidores Node.js. Vamos explorar em mais detalhes como esse processo funciona no contexto do TypeScript.

- Código TypeScript: as pessoas desenvolvedoras escrevem seu código-fonte em TypeScript, usando a sintaxe e os recursos fornecidos pelo TypeScript, como a declaração de tipos.
- Arquivo de origem: o código TypeScript é armazenado em um ou mais arquivos com extensão .ts ou .tsx (se você estiver trabalhando com React).
- Compilador TypeScript (tsc): o TypeScript fornece um compilador de código-fonte chamado tsc, que é uma ferramenta de linha de comando que transforma o código TypeScript em JavaScript. Os(as) desenvolvedores(as) normalmente executam o tsc manualmente ou configuram ferramentas de compilação automatizada em seus ambientes de desenvolvimento.
- Configuração do compilador: ele é configurado por meio de um arquivo chamado tsconfig.json, que define as opções de compilação, como o destino

(versão do JavaScript), o local de saída dos arquivos JavaScript gerados e outras configurações relacionadas ao projeto.

- Verificação de tipos: durante a compilação, o TypeScript realiza a verificação de tipos. Isso envolve a análise do código-fonte em busca de erros de tipo, incompatibilidades e outros problemas relacionados a tipos. Se encontrados, o compilador os destacará e impedirá a geração do código JavaScript até que eles sejam corrigidos.
- Geração de código JavaScript: após verificar os tipos e garantir que não há erros, o TypeScript cria um ou mais arquivos JavaScript que funcionam da mesma forma que o código TypeScript, mas são escritos em JavaScript simples.
- Arquivos de saída: os arquivos JavaScript gerados podem ser configurados para serem salvos em um diretório específico de saída, conforme definido no arquivo tsconfig.json.
- Execução: o código JavaScript gerado pode ser executado em qualquer ambiente JavaScript compatível, como um navegador, um servidor Node.js ou até mesmo em dispositivos IoT que suportam JavaScript.

A compilação é uma etapa crucial para garantir que o código TypeScript seja compatível com o ambiente JavaScript de destino. Ela permite que as pessoas desenvolvedoras aproveitem os recursos de tipagem estática e outras vantagens do TypeScript durante o desenvolvimento, enquanto ainda podem executar o código resultante em qualquer lugar onde JavaScript seja suportado.

A compilação desempenha um papel fundamental ao garantir que o código TypeScript funcione sem problemas em ambientes JavaScript. No entanto, para que a compilação seja eficaz, é essencial que os(as) desenvolvedores(as) compreendam os conceitos de tipos explícitos e implícitos. Vamos explorar esses conceitos em mais detalhes para entender como eles influenciam o processo de compilação e a qualidade do código TypeScript resultante.

Declarações de variáveis com Tipos Explícitos e Implícitos

A declaração de variáveis com tipos explícitos e implícitos refere-se à maneira como especificamos o tipo de uma variável em TypeScript.

Variáveis com Tipos Explícitos

Ao declarar uma variável com um tipo explícito, você indica explicitamente qual tipo de valor a variável pode armazenar. Isso é feito usando a seguinte sintaxe:

```
let idade: number;
idade = 30 // Valor atribuído é do tipo number
```

Neste exemplo, a variável idade é declarada com um tipo explícito number, o que significa que só pode armazenar valores numéricos. Qualquer tentativa de atribuir um valor de um tipo diferente resultará em um erro de compilação.

Variáveis com Tipos Implícitos

Em contrapartida, você também pode declarar variáveis sem especificar explicitamente um tipo. O TypeScript pode inferir automaticamente o tipo com base no valor atribuído à variável:

```
let nome = "João" // TS infere que o 'nome' é do tipo string
```

Nesse caso, o TypeScript inferiu que a variável nome é do tipo string porque inicializamos com uma string. Isso é chamado de "tipagem implícita". No entanto, observe que, uma vez que o tipo é inferido, você não pode atribuir um valor de um tipo diferente a essa variável.

Em resumo, podemos escolher declarar variáveis com tipos explícitos, onde especificamos diretamente o tipo, ou podemos deixar o TypeScript inferir o tipo com base no valor atribuído, usando a tipagem implícita. Ambos os métodos têm seu lugar no desenvolvimento TypeScript, dependendo das necessidades do nosso código e da clareza que desejamos fornecer ao leitor ou a outros desenvolvedores. No entanto, para entender como essa escolha afeta nossos programas, é importante compreender

os tipos básicos disponíveis e como usá-los efetivamente. Portanto, vamos agora explorar em detalhes os tipos básicos em TypeScript: number, string e boolean.

Tipos Básicos em TypeScript

Em TypeScript, existem alguns tipos básicos que são fundamentais para o desenvolvimento de código. Esses tipos incluem:

number: representa números inteiros ou de ponto flutuante. Exemplo:

```
const idade: number = 30;

const altura: number = 1.75;
```

string: representa sequências de caracteres, como texto. Exemplo:

```
const nome: string = "Thamiris";

const mensagem: string = "Olá, Thamiris!";
```

boolean: representa valores lógicos verdadeiro (true) ou falso (false). Exemplo:

```
const ativo: boolean = true;

const isAdmin: boolean = false;
```

Esses tipos básicos são úteis para declarar e definir variáveis que armazenam diferentes tipos de dados em um programa TypeScript. Quando utilizamos esses tipos, o TypeScript pode realizar a verificação de tipos em tempo de compilação, ajudando a evitar erros durante a execução do código.

É importante notar que o TypeScript permite que seja atribuído um valor de um tipo diferente a uma variável, desde que o novo valor seja compatível com o tipo declarado. Por exemplo, podemos reatribuir uma variável number com outro valor

number, mas não com uma string ou boolean, a menos que você faça uma conversão explícita de tipo.

```
const x: number = 5;

x = 10; // Isso é válido

x = "abc"; // Isso geraria um erro, pois "abc" não é um número
```

Os tipos básicos em TypeScript são a base para criar tipos mais complexos, como objetos, arrays e tipos personalizados. Eles desempenham um papel fundamental na tipagem estática da linguagem, tornando o código mais seguro e compreensível.

Arrays em TypeScript

Em TypeScript, assim como em outras linguagens de programação, os arrays são tipos de dados que permitem armazenar uma coleção de valores, sejam eles números, strings, objetos ou qualquer outro tipo de dado. Os arrays são uma estrutura de dados fundamental e flexível.

Declarando e manipulando um Array

Podemos criar um Array usando a notação de colchetes []:

```
let numeros: number[] = [1, 2, 3, 4, 5];
let nomes: string[] = ["Maria", "João", "Samuel"];
```

Para acessar esses elementos, usamos índices numéricos que se iniciam em 0:

```
let primeiroNumero = numeros[0]; // Obtém o primeiro número (1)
let segundoNome = nomes[1];      // Obtém o segundo nome ("João")
```

Para manipular arrays, o TypeScript oferece uma variedade de métodos, como push, pop, shift. Abaixo, temos uma pequena lista com alguns métodos e suas funcionalidades.

- Pop(): remove o último elemento de um array e retorna esse elemento.
- Push(): adiciona um ou mais elementos ao final de um array e retorna o tamanho desse array.
- Shift(): remove o primeiro elemento de um array e retorna esse elemento.
- Unshift(): adiciona um ou mais elementos no início de um array e retorna o tamanho deste array.
- reduce(): executa uma função reducer (fornecida por você) para cada elemento do array, resultando num único valor de retorno.
- reverse(): inverte os itens de um array. O primeiro elemento do array se torna o último e o último torna-se o primeiro.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, aprendemos sobre TypeScript, um superset de JavaScript, que adiciona tipagem estática e traz diversas características, como segurança e melhor ferramenta de desenvolvimento. Vimos como a compilação é crucial para transformar o código TypeScript em JavaScript executável em navegadores e servidores Node.js. Além disso, exploramos a declaração de variáveis com tipos explícitos e implícitos, destacando como esses conceitos afetam a clareza e a robustez do código TypeScript.

REFERÊNCIAS

ADRIANO, T. **Guia prático de Typescript: Melhore suas aplicações JavaScript**. São Paulo: Casa do Código, 2021.

RABELO, E. **TypeScript: Entendendo a notação de tipos**. Disponível em: <https://oieduardorabelo.medium.com/typescript-entendendo-a-nota%C3%A7%C3%A3o-de-tipos-9e8c1c89ef62>. Acesso em: 24 jan. 2024.

RABELO, E. **TypeScript: O guia definitivo**. Disponível em: <https://oieduardorabelo.medium.com/typescript-o-guia-definitivo-1a63b04259cc>. Acesso em: 24 jan. 2024.

PALAVRAS-CHAVE

Palavras-chave: Typescript. Desenvolvimento Básico. Superset.