# **Linguagem de Programação: TypeScript**

**1. Introdução**

TypeScript é uma linguagem de programação de tipos estáticos que é compilada para JavaScript. É uma extensão de JavaScript que adiciona tipos ao código para ajudar a prevenir erros e tornar o desenvolvimento mais fácil. O TypeScript foi criado pelo time do Microsoft e tornou-se open-source em 2012. Desde então, tornou-se amplamente adotado pelos desenvolvedores por sua capacidade de ajudar a escrever códigos mais confiáveis e de fácil manutenção. O TypeScript é uma excelente escolha para os desenvolvedores que procuram uma linguagem de programação mais segura e confiável do que o JavaScript puro.

**2. Lexemas**

Em TypeScript, existem algumas regras que você precisa seguir ao criar identificadores, ou seja, nomes para variáveis, funções, classes, interfaces, etc.:

1. O primeiro caractere deve ser uma letra, um sublinhado ( \_ ) ou um sinal de dólar ($).
2. Os caracteres subsequentes podem ser letras, números, sublinhados ou sinais de dólar.
3. Os identificadores não podem ser palavras-chave ou palavras reservadas do TypeScript.
4. Os identificadores são sensíveis a maiúsculas e minúsculas, ou seja, "nome" e "Nome" são identificadores diferentes.
5. É uma boa prática usar nomes significativos e descritivos para os identificadores, evitando usar nomes curtos e genéricos como "temp" ou "val".

Essas regras se aplicam tanto ao código TypeScript quanto ao JavaScript, pois o TypeScript é uma camada de tipagem adicional sobre o JavaScript.

**2.1 Comentários**

Em TypeScript, os comentários são usados para adicionar notas explicativas ao código. Eles são ignorados pelo compilador e não afetam a execução do código. Existem duas formas de adicionar comentários ao código TypeScript: comentários de linha única e comentários de múltiplas linhas.

Comentários de linha única são adicionados usando duas barras (//) e vão até o final da linha:

// Este é um comentário de linha única

Comentários de múltiplas linhas são adicionados usando uma barra seguida de um asterisco (/) no início e um (\*) seguido de uma barra (/) no final:

/\*

Este é um

comentário de

múltiplas linhas

\*/

**2.2 Palavras reservadas**

As palavras-chave em TypeScript são aquelas que têm um significado especial e não podem ser usadas como nomes de variáveis, funções ou outros identificadores. Palavras-chave que podemos encontrar em TypeScript:

* break: usado para sair de um laço
* case: usado em um switch statement
* catch: usado para capturar exceções
* class: usado para definir uma classe
* const: usado para declarar uma constante
* continue: usado para pular para a próxima iteração de um laço
* debugger: usado para parar a execução do código e entrar no depurador
* default: usado como um caso padrão em um switch statement
* delete: usado para excluir uma propriedade de um objeto
* do: usado para iniciar um laço do-while
* else: usado como uma cláusula de ramificação em uma estrutura de controle de fluxo
* enum: usado para definir um tipo de dados enumerado
* export: usado para exportar uma classe, uma função ou um valor de um módulo
* extends: usado para estender uma classe
* finally: usado para garantir que um bloco de código será executado, independentemente de uma exceção ser capturada ou não
* for: usado para iniciar um laço for
* function: usado para declarar uma função
* if: usado para testar uma condição
* implements: usado para implementar uma interface
* import: usado para importar uma classe, uma função ou um valor de outro módulo
* in: usado para testar se um determinado valor existe em uma array ou objeto
* instanceof: usado para testar se um objeto é uma instância de uma determinada classe ou interface
* interface: usado para definir uma interface
* let: usado para declarar uma variável
* new: usado para criar uma nova instância de uma classe
* null: um valor especial que representa ausência de valor
* return: usado para retornar um valor de uma função
* super: usado para acessar a classe pai em uma classe filha
* switch: usado para executar diferentes ações com base em uma variável
* this: usado para acessar o objeto atual
* throw: usado para lançar uma exceção

**2.3 Literais reservados**

As palavras reservadas do TypeScript incluem:

* any
* never
* number
* boolean
* string
* symbol
* undefined
* null
* void
* this
* let
* const
* class
* extends
* implements
* interface
* namespace
* module
* enum
* type
* as
* private
* protected
* public
* static
* readonly
* constructor
* in
* of
* from
* try
* catch
* switch
* case
* break
* continue
* default
* return
* do
* while
* for
* function
* if
* else
* with
* new
* delete
* typeof
* instanceof
* await
* yield

**2.4 Operadores e delimitadores**

Os operadores e delimitadores do TypeScript incluem:

Operadores:

* Arithmetic operators (+, -, \*, /, %, ++, --)
* Relational operators (>, >=, <, <=, ==, !=, ===, !==)
* Logical operators (&&, ||, !)
* Bitwise operators (&, |, ^, ~, <<, >>, >>>)
* Assignment operators (=, +=, -=, \*=, /=, %=, <<=, >>=, >>>=, &=, |=, ^=)
* Ternary operator (?:)
* Spread operator (...)
* Union operator (|)
* Intersection operator (&)
* Type-guards operator (as)
* In operator (in)
* instanceof operator (instanceof)

Delimitadores:

* Parentheses (())
* Brackets ([] and [])
* Braces ({})
* Comma (,)
* Semicolon (;)
* Period (.)
* Question mark (?)
* Colon (:)
* Equal sign (=)
* Plus sign (+)
* Minus sign (-)
* Asterisk (\*)
* Forward slash (/)
* Percent sign (%)
* Tilde (~)
* Exclamation point (!)
* Ampersand (&)
* Pipe (|)
* Caret (^)
* Backtick (`)
* Double quotes (")
* Single quotes (')

**2.5 Literais String**

No TypeScript, existem duas maneiras de representar literais de string: com aspas simples (' ') ou com aspas duplas (" ").

let integerLiteral = 100;

let decimalLiteral = 3.14;

Além disso, o TypeScript suporta notação científica, que permite representar números grandes ou muito pequenos de maneira concisa.

let scientificNotation1 = 5e10;

let scientificNotation2 = 1.23456e-6;

**2.6 Literais Numéricos**

Existem dois tipos de literais numéricos: inteiros e decimais.

**2.6.1 Literais Integer**

Em TypeScript, um literal inteiro é uma constante numérica sem ponto flutuante. Existem duas formas de representar literais inteiros: decimal e hexadecimal.

**2.6.1.1 Literal Decimal**

Esses são representados por uma sequência de dígitos decimais, sem qualquer prefixo ou sufixo. Por exemplo:

let age = 30; let quantity = 10;

**2.6.1.2 Literal Hexadecimal**

Literais inteiros hexadecimais: Esses são representados por uma sequência de dígitos hexadecimais precedidos pelo prefixo "0x". Por exemplo:

let color = 0xff0000; let value = 0x1a;

**2.6.4 Literais de ponto flutuante**

Em TypeScript, um literal de ponto flutuante é uma constante numérica com casas decimais. Eles são representados por uma sequência de dígitos, com um ponto decimal opcional para separar a parte inteira da parte decimal.

Aqui estão alguns exemplos de literais de ponto flutuante em TypeScript:

let price = 19.99;

let interestRate = 0.05;

let pi = 3.14159265359;

Você também pode usar notação científica para representar números grandes ou pequenos:

let distance = 1.23e6; // 1.23 \* 10^6

let mass = 5.67e-8; // 5.67 \* 10^-8

**2.6.5 Literais Booleanas**

No TypeScript, existem dois tipos de literais booleanas:

let isValid = true;

let isInvalid = false;

Esses literais são comumente usados em expressões lógicas, como em condicionais e loops, para determinar se uma determinada condição é verdadeira ou falsa.

**2.6.6 Literais Char**

Em TypeScript, um literal de caractere (também conhecido como "caractere literal") é uma constante de um único caractere. Eles são representados por uma única citação simples (') ou aspas duplas (") que envolvem o caractere.

Aqui estão alguns exemplos de literais de caractere em TypeScript:

let letter = 'A';

let symbol = '#';

let newline = '\n';

Você também pode representar caracteres especiais usando a notação de escape ():

let quote = '\''; // single quote

let doubleQuote = '\"'; // double quote

let backslash = '\\'; // backslash

**Referências:**

https://www.typescriptlang.org/pt/