Typescript – principais comandos

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Comentário

```
//comentário de linha

/**comentário
 * de
 * bloco
 */
```

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Saída

- Saída navegador

```
console.Log("olá");
```

- Saída modal

```
alert("olá");
```

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Estrutura de uma classe

```
import { Component } from '@angular/core';

export class HomePage {
    teste : string;

    emitirMensagem(nome: string) : void{
        alert('olá '+nome);
    }

    hello() : string {
        let retornar= 'olá';
        return retornar;
    }
}
```

Variáveis

- Tipo primitivo

  [let] *nome_variavel* : tipo = valor;

  – Exemplo

  ```
  let var1 : boolean = true;   [true | false]
  let var2 : number = 55;
  let var3 : string = "POO";
  ```

- Any

  – assumir qualquer valor.

  ```
  let variavel : any = 4; [10.2 | 0x001]
  variavel = "agora é uma string";
  variavel = false; //agora é um boolean
  ```

number

- Exempo

```
let decimal: number = 6;
let hex: number = 0xf00d;
let binary: number = 0b1010;
let octal: number = 0o744;
```

string

- **Exemplo**

```
let nome: string = "Joaquim Silva";
nome = 'Joaquim Silva Júnior';

let imprimir = "meu nome é " + nome;

console.log(imprimir);
```

Expressões embutidas

- Por peio do acento grave (`), podemos quebrar linhas ou inserir variáveis

```
let a: string = `
    olá meu nome né ${nome}
    e estou estudando POO
`;
```

É possível quebrar linhas e inserir variáveis

Expressões embutidas

- **Exemplo**

```
let nome: string = "Joaquim Silva";

let imprimir = "meu nome é " + nome;

imprimir = `meu nome é ${nome} `;

console.log(imprimir);
```

Expressões embutidas

- **Exemplo**

```
let imprimir = `a soma de 5 + 5  é ${ 5 + 5} `;

console.log(imprimir);
```

Operadores

Operador	Nome	Descrição	Exemplo	Resultado
x+y	adição	Soma x e y	3+2	5
x-y	subtração	Diferença de x e y	12-2	10
x*y	multiplicação	Multiplica x e y	2*10	20
x/y	divisão	Divisão de x and y	30/5	6
x%y	mod	Resto da divisão de x e y	7%3	1
-x	negação	Negativo de x	-9	-9
x.y	concatenar	Concatena x e y	"programação "."POO"	programação POO

Conversão

- Conversão string para int

  Number('123');

  parseInt('123');

  parseFloat('123.45');

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Estrutura de Decisão

```
if(comparação){
    //comando um
    //comando dois
    //...
}else{
    //comando um
    //comando dois
    //...
}
```

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Operadores

- **Relacionais**

  >, <, >=, <=, !=, ==, ===

- **Lógicos**
  - **&& (And), || (OR), ! (NOT)**

```
let num1: number = 30;
let num2: number = 3;
let valida1 = num1 == num2;
let valida2 = num1 > num2;
let valida3 = num1 < num2;
console.log(valida1);
console.log(valida2);
console.log(valida3);
```

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Alert

```
let num : number = -55;
if(num > 0){
     alert("é positivo");
} else {
     alert("é negativo");
}
```

Operador Ternário

- Frequentemente usado como um atalho para a instrução “if”;
- Sintaxe

  Condition ? result1 : result2;

- Exemplo

```
let num : number = 7;
let resultado : string = num > 0 ? "positivo"  : "negativo";
```

Operador Ternário

- **Exemplo**

```
let num : number = 7;
let resultado : string = num > 0 ? "positivo"  : "negativo";
alert(resultado);
```

- **Equivalente em if**

```
let num : number = 7;
let resultado : string;
if( num > 0){
      resultado = "positivo";
} else {
      resultado = "negativo";
}
alert(resultado);
```

Estrutura de Repetiação

- while
  - Para executar os comandos, necessariamente todas as interações irão passar pela condição;
- do while
  - A primeira interação não irá passar pela condição – ao menos uma vez irá ser executada
- for
  - Esta estrutura determina um contador para ser utilizado
- for (foreach)
  - Efetua a interação na coleção em a utilização do contador

while

- Sintaxe

```
while(condition) {
    // comandos
}
```

- Exemplo

```
let num:number = 5;
let fatorial:number = 1;
while(num >=1) {
    fatorial = fatorial * num;
    num--;
}
console.log("O fatorial é "+fatorial);
```

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

do-while

- **Sintaxe**

```
do {
    // comandos
} while(condition)
```

- **Exemplo**

```
let i = 0;
do {
    i += 1;
    console.log(i);
} while (i < 5);
```

for

- **Sintaxe**

```
for(contador; condição; incremento) {
    // comandos
}
```

- **Exemplo**

```
let num:number = 5;
let i:number;
let fatorial = 1;
for(let i = num;i>=1;i--) {
    fatorial *= i;
}
console.log(fatorial);
```

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

foreach

- **Sintaxe**

```
for (var val in list) {
    //comandos
}
```

- **Exemplo**

```
let j:string;
let n:string[] = ["Jorge","Bianca","Maycon"];
for(j in n) {
    console.log(n[j])
}
```

break

- **Sintaxe**

  break

- **Exemplo**

```
let continua: string = "s";
let contador: number = 0;
do {
    console.log(contador);
    contador = contador + 1;
    if (contador > 5) {
        break;
    }
} while (continua="s");
```

Vetor

- vetor

  [let] *nome_variavel* : tipo[] = [valor1, valor2, ... , valorX];

  – Exemplo

  let numeros: number[] = [1,2,3]

- Any

  – assumir qualquer valor.

  let variavel : any[] = [15, "azul", false];

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Vetor

- **Exemplo**

```
let nomes: string[] = ['Joaquim', 'Melissa', 'Zeca'];
nomes[2] = 'Paulina';

for(let i = 0; i < nomes.length; i++) {
    console.log(nomes[i]);
}
```

Vetor

- **Exemplo**

```
let numeros: number[] = [10,20,30];
numeros[2] = 30;

for(let i; item of nomes) {
    console.log(item);
}
```

Vetor

- Sintaxe

  nomeVar = new Array<tipo>();

- Exemplo

  frutas = new Array<string>();

  frutas.push("maçã");

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Vetor

- Exemplo

```
frutas = new Array<string>();
frutas.push("maçã");
frutas.push("morango");
frutas.push("manga");
frutas.splice(0,1);                  //remove maçã
frutas.pop();                        //remove o último elemento
frutas.indexOf("morango"); //retorna 1
```

Gerar Número Aleatório

- Gerando número aleatório

```
let val = Math.floor(Math.random() * 10);
```

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br

Leitura Complementar

- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

- https://www.typescriptlang.org/docs/handbook/basic-types.html

Atividade

1. Gere um número aleatório e informe se o número é positivo ou negativo;
2. Gere um número aleatório e informe se o número é par ou ímpar;
3. Gere um número aleatório e informe se o número é par e positivo;
4. Gere 3 números aleatórios e informe qual é o maior e qual é o menor;
5. Imprima números pares de 0 a 1000;
6. Imprima número que terminam com 5 de 0 a 1000;
7. Defina um ano de nascimento e informe a idade;

Hélio Toshio Kamakawa – helio.kamakawa@ifpr.edu.br