TYPESCRIPT

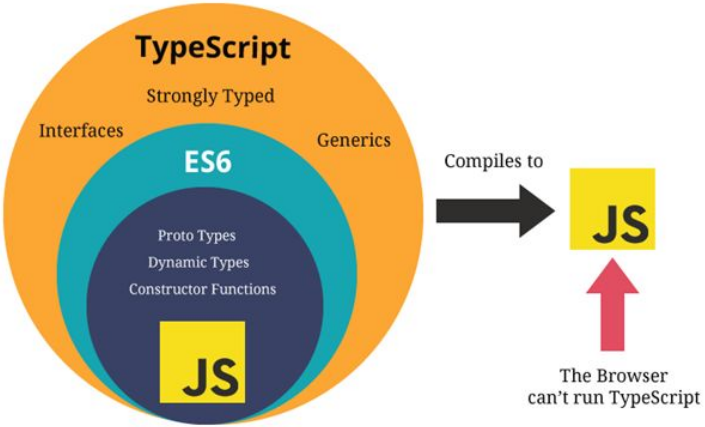
Linguagem de programação baseada em javascript para servidores

SAE COMP

INTRODUÇÃO AO TYPESCRIPT

A linguagem

Typescript é um sistema de tipo s para o javascript criado pela microsoft. Eles não são tipos de verdade, ou seja, em runtime eles não são impostos, porém ajudam muito durante o desenvolvimento e debugging.



Variáveis

O typescript tem dois tipos de variáveis principais: const e let. A const é uma variável imutável de escopo e a let pode ser mudada também dentro do seu escopo. Sempre que possível é melhor usar const para preservar memória.

```
1  const var1 = 10;
2
3  let var2 = 20;
```

Objetos e listas

Objetos são muito utilizados dentro do typescript para representar dados. Eles são um hash map que podem ter chaves que acessam cada informação. As listas ficam dentro de [] e podem ser indexadas.

```typescript
const obj1 = {
    name: "Lucas",
    age: 23,
    isStudent: true,
    hobbies: ["programming", "reading"],
    address: {
        street: "Rua A",
        number: 123
    }
}

console.log(obj1.name); // Lucas
console.log(obj1.hobbies[0]); // programming
```

Funções

As funções servem para que quando chamadas elas rodem um pedaço de código específico. Elas podem ter argumentos de entrada e podem ser definidas com um nome ou como anônimas.

```typescript
function soma(a: number, b: number): number {
    const soma: number = a + b;
    return soma;
}

// função anônima
const concat_str = (str1: string, str2: string): string => {
    const str: string = str1 + str2;
    return str;
}

// função anônima passando como parâmetro
[1, 2, 3].forEach((item: number) => {
    console.log(item);
});
```

Tipos

Os tipos são definidos colocando dois pontos depois das variáveis. Eles são exclusivos do typescript e não funcionam em runtime.

```typescript
// booleano
let isDone: boolean = false;
// números
let decimal: number = 6;
// textos
let color: string = "blue";
// listas
let list: number[] = [1, 2, 3];
// enum
enum Color {
    Red,
    Green,
    Blue,
}
let c: Color = Color.Green;
```

```typescript
// any
let notSure: any = 4;
notSure = "string";
// juntando tipos
let strOrNum: string | number;
// null e undefined
let u: number | undefined;
let n: null | number;
```

Interfaces

Quando se deseja fazer um tipo mais complexo e reutilizável, se usa a interface do typescript. Nela, se define tipos de objetos e ela pode ser usada como se fosse um tipo comum.

```typescript
interface IAddress {
    street: string;
    number: number;
}

interface IStudent {
    name: string;
    age: number;
    isStudent: boolean;
    hobbies: string[];
    address: IAddress;
}
```

```typescript
const student: IStudent = {
    name: "Lucas",
    age: 23,
    isStudent: true,
    hobbies: ["programming", "reading"],
    address: {
        street: "Rua A",
        number: 123
    }
}
```

if e loops

As estruturas padrão do typescript são muito parecidas com as do C. Uma diferença é que no if, existe dois tipos de igual: == e ===.

```ts
const str = "1";

// if sem type assertion
if (str == 1) { // vai retornar verdadeiro
}

if (str === 1) { // vai retornar falso
}

if (str === "1") { // vai retornar verdadeiro
}
```

```ts
// loop com for
for (let index = 0; index < 10; index++) {

}

// loop com while
while (str === "1") {

}
```

Métodos de arrays

Como tudo dentro do typescript é um objeto, podemos chamar propriedades que realizam operações.

```ts
// map: itera sobre o array e retorna cada elemento da função
["a", "b", "c"].map(el => el.toUpperCase());
// retorna ["A", "B", "C"]

// filter: remove itens que se não encaixam na condição
[1, 2, 3, 4].filter(el => el % 2 === 0);
// retorna: [2, 4]

// reduce: acumula valores em acc e retorna
[10, 20, 30].reduce((acc, el) => acc + el, 0);
// retorna: 60

// includes: verifica se existe o valor
["a", "b", "c"].includes("a")
// retorna true
```

```ts
// find: retorna a primeira ocorrência da condição
const users = [{ id: 1 }, { id: 2 }];
users.find(u => u.id === 2);
// retorna { id: 2 }

// forEach: itera sobre o array executando a função
[1, 2, 3].forEach(el => console.log(el));

// every: retorna true se todos os itens encaixam na condição
[2, 4, 6].every(n => n % 2 === 0);
// retorna true

// some: retorna true se pelo menos um item se encaixa na condição
[1, 3, 4].some(n => n % 2 === 0);
// retorna true
```

Métodos de objetos

Como tudo dentro do typescript é um objeto, podemos chamar propriedades que realizam operações.

```ts
// retorna as chaves do objeto
Object.keys({ a: 1, b: 2 }); // ["a", "b"]

// retorna os valores das chaves
Object.values({ a: 1, b: 2 }); // [1, 2]


//retorna uma lista de tuplas com as chaves e valores
Object.entries({ a: 1, b: 2 }); // [["a",1], ["b",2]]

// verifica se existe a chave no objeto
"a" in { a: 1, b: 2 }; // true

// apaga uma chave do objeto
const obj = { x: 10 };
delete obj.x; // obj agora é {}
```

INSTALANDO O NODE JS

Rodando JS e TS

O node.js permite executar programas javascript direto no computador, sem a necessidade de ser um site no navegador. Para instalar entre em https://nodejs.org/en/download/

```
# executa o arquivo main.ts
npx tsx main.ts

# compila o arquivo main.ts para main.js
npx tsc main.ts

# executa o arquivo main.js
node main.js
```

ATIVIDADES

Atividade 1

Fazer um sistema de cadastro de tarefas

Crie uma aplicação em TypeScript que permita:

- Adicionar uma tarefa
- Listar todas as tarefas
- Marcar uma tarefa como concluída
- Filtrar tarefas por status (pendente, concluída)

Requisitos:

- Usar uma interface Tarefa com id, titulo, status
- Usar um enum Status com Pendente e Concluida
- Criar funções separadas: adicionarTarefa, listarTarefas, concluirTarefa, filtrarPorStatus

MUITO OBRIGADO


@saecomp.ec


saecomp@usp.br


saecomp.github.io


Prédio da Engenharia de Computação,
Campus 2, USP São Carlos

SAECOMP