Relatório 3 - Prática: Iniciando com TypeScript (II)

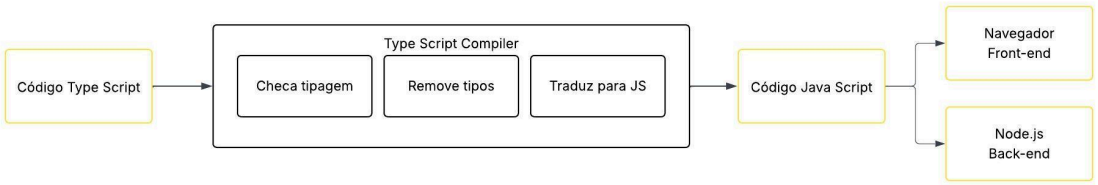
José Carlos Seben de Souza Leite

Descrição da atividade

Dando início ao primeiro card pratico do fast camp, nossa primeira tarefa é acompanhar o vídeo "**Curso de TypeScript para Completos Iniciantes**" onde no mesmo começamos com a parte teórica de diferença entre o JavaScript e o TypeScript e a primeira diferença citada são os tipos de linguagens dinâmicas ou estáticas, a diferença dessas se baseia em linguagens que utilizam variáveis dinâmicas como, por exemplo, dessas linguagens o próprio **JavaScript** aqui uma variável pode ser declarada sem tipo e depois definirmos um tipo em específico para ela, por exemplo, um número e no decorrer do código podemos mudar o tipo dessa mesma variável se houver necessidade, por exemplo, mudando de um número como havia sido definida para uma string, já nas linguagens que utilizam variáveis estáticas como exemplo dessas o **TypeScript** ao criamos uma variável já precisamos definir a tipagem da mesma e independente de o que aconteça não podemos alterar a tipagem desta mesma variável no decorrer de código.

O que é TypeScript?

TypeScript é uma linguagem de programação criada com base em JavaScrip basicamente criada por cima do JavaScript com o intuito de transformar de uma linguagem dinâmica para uma linguagem estática e adicionar diversas novas funcionalidades ao JavaScript, tanto que quando compilamos um código em TypeScript que não possui erros faz a conversão para sua versão equivalente em JS pelo compilador Type Script Compiler, essa conversão que permite que esta linguagem possa ser utilizada tanto para o back-end quanto para o front-end.

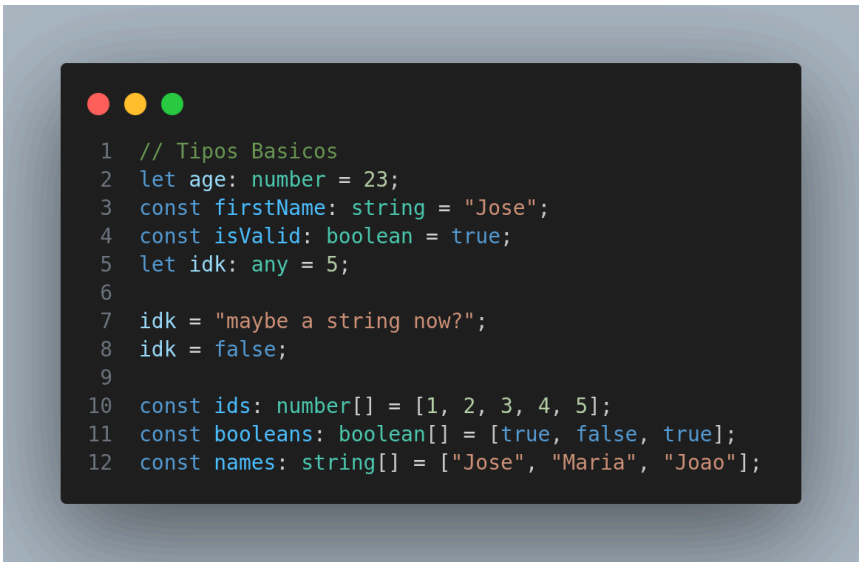


No decorrer do vídeo nos é também mostrado o porquê utilizar o Type Script como no exemplo utilizado fazendo os códigos em TS podemos utilizar de ferramentas modernas do JS para facilitar no desenvolvimento e quando o sistema for compilado para JS ele ficara adaptado para navegadores mais antigos, então utilizando o TS poupamos tempo e graças a sua compilação para JS também prevenimos erros que poderia dar se tivéssemos utilizado diretamente o JS assim garantindo compatibilidade com o máximo de navegadores e nodes possíveis.

Partindo para a parte prática após a explicação de download do node e do TS vai ser uma explicação sobre variáveis sendo elas.

Codigos contidos em:
https://github.com/SebenJose/Fast-Camp---Web/blob/main/Card%203%20-%20Pr%C3%A1tica%3A%20Iniciando%20com%20TypeScript%20(II)/Codigos%20Aula/src/index.ts

Variáveis padrões.



```
// Tipos Basicos
let age: number = 23;
const firstName: string = "Jose";
const isValid: boolean = true;
let idk: any = 5;

idk = "maybe a string now?";
idk = false;

const ids: number[] = [1, 2, 3, 4, 5];
const booleans: boolean[] = [true, false, true];
const names: string[] = ["Jose", "Maria", "Joao"];
```

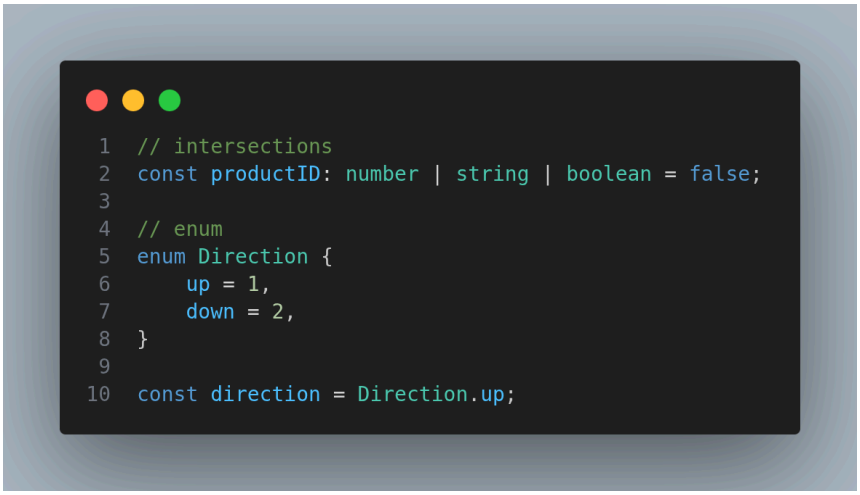
Como números, strings e booleanos e o “any” que é um tipo de variável do TS que pode receber qualquer tipo por isso ela leva o tipo qualquer, e temos também as listas que são listas de variáveis de um mesmo tipo.

Tuplas e Lista de Tuplas.

```
// Tuplas
const person: [number, string] = [20, "jane"];

// Lita de Tuplas
const people: [number, string][] = [
    [10, "alice"],
    [20, "bob"],
    [30, "charlie"]
]

```

Tuplas são basicamente uma lista personalizada onde na mesma podemos colocar mais de um tipo de variável na lista como esta no exemplo number e string a condição que ao inserir os valores dentro da lista seja respeitada a ordem dos tipos como está primeiro o number depois a string se na declaração dos valores primeiro você declarar a string vai dar erro, e a lista de tuplas é basicamente o que diz o nome você cria uma lista com diversas tuplas.

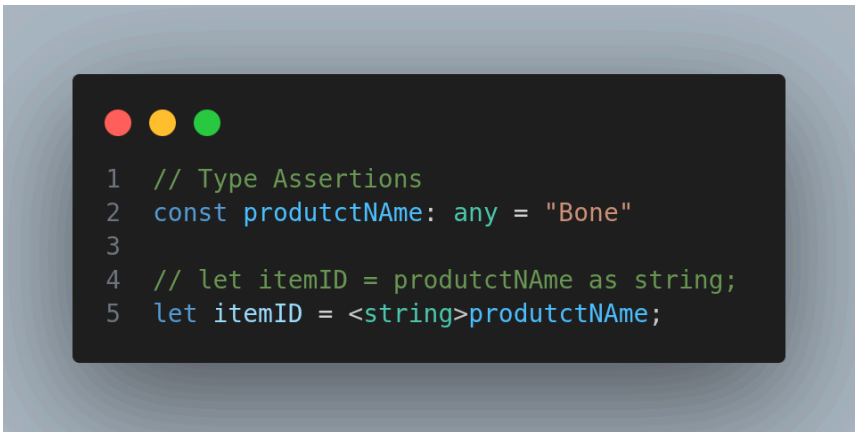
Union e Enum.



```
// intersections
const productID: number | string | boolean = false;

// enum
enum Direction {
    up = 1,
    down = 2,
}

const direction = Direction.up;
```

Aqui acabou havendo uma confusão, pois no vídeo foi apresentado este exemplo do productID como uma inserction que é quando uma variável é um tipo ou outro utilizando o &, mas isso é utilizado para objetos nesse caso o que realmente é uma união onde uma variavel pode receber vários tipos que ou ela vai ser um numerou ou uma string, ou um booleano diferente do caso do inserction onde ela supostamente iria ser um número e uma string ao mesmo tempo, agora sobre os ununs eles nos permitem criar conjuntos de constantes nomeadas como no exemplo acima onde up é 1 e down é o 2 e logo em baixo um exemplo de como utilizaremos, por exemplo, ao invés de usar 1 utilizamos o up que vai dar na mesma deixando o código mais legível e salvando de precisar ficar lembrando o número de cada coisa.

Type Assertions.

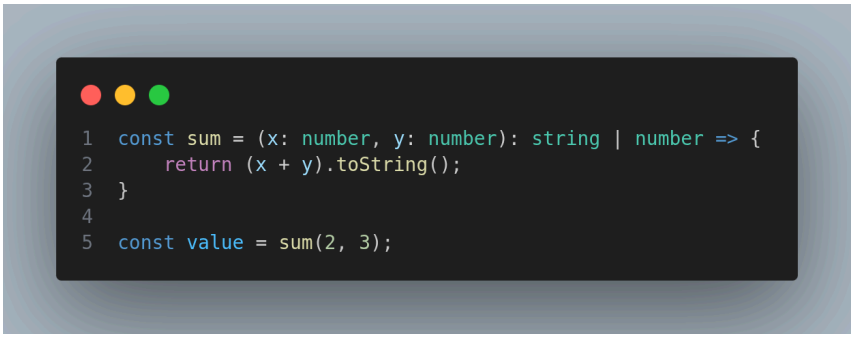


```
// Type Assertions
const produtctNAme: any = "Bone"

// let itemID = produtctNAme as string;
let itemID = <string>produtctNAme;
```

Type assertions servem basicamente para alterarmos o tipo de uma variável como no exemplo a cima onde ela foi declarada como qualquer e logo abaixo a definimos como uma string.

Funções

Códigos contidos em:
https://github.com/SebenJose/Fast-Camp---Web/blob/main/Card%203%20-%20Pr%C3%A1tica%3A%20Iniciando%20com%20TypeScript%20(II)/Codigos%20Aula/src/functions.ts



```ts
const sum = (x: number, y: number): string | number => {
    return (x + y).toString();
}

const value = sum(2, 3);
```

Este foi o primeiro exemplo a ser mostrado inicialmente foi feito apenas usando numbers como tipos de variáveis para mostrar como funcionava a função de fazer uma soma, depois foi modificado os tipos para string para nos mostrar como o TS é inteligente e se adapta bem as mudanças de tipos e prevenções de erros.

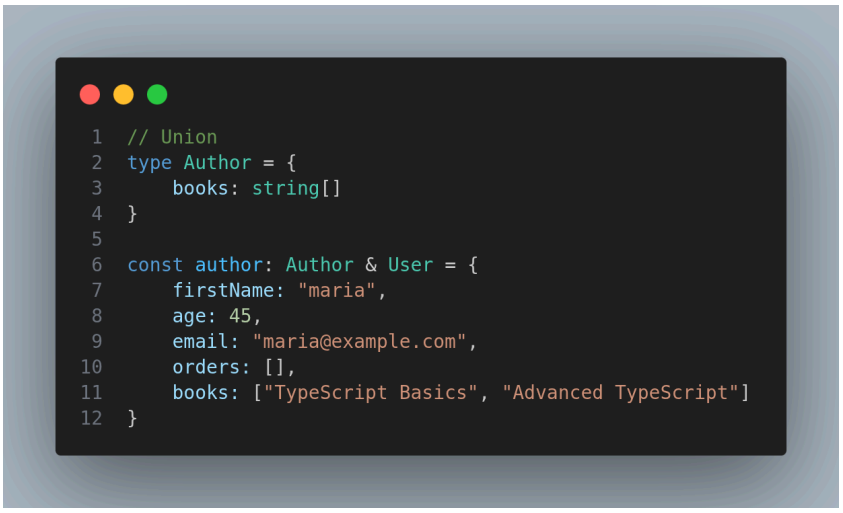
Objetos

Códigos contidos em:
https://github.com/SebenJose/Fast-Camp---Web/blob/main/Card%203%20-%20Pr%C3%A1tica%3A%20Iniciando%20com%20TypeScript%20(II)/Codigos%20Aula/src/objects.ts

```ts
// type
type Order = {
    productId: string
    price: number
}

type User = {
    firstName: string
    age: number
    email: string
    password?: string
    orders: Order[]
};

const user: User = {
    firstName: "jose",
    age: 30,
    email: "jose@example.com",
    password: "secret123",
    orders: [{ productId: "abc123", price: 29.99 }]
}

const printLog = (message: string): void => {}

printLog(user.password!)
```

Entrando agora na explicação de objetos, de como funcionam e de coisas que podemos estar fazendo como, por exemplo, ao criar o objeto podemos usar o “?” para definir um valor como não sendo obrigatório, e também utilizando o “!” para quando formos usar esse valor marcado como não obrigatório para basicamente afirmar que esse valor mesmo não sendo obrigatório existe.

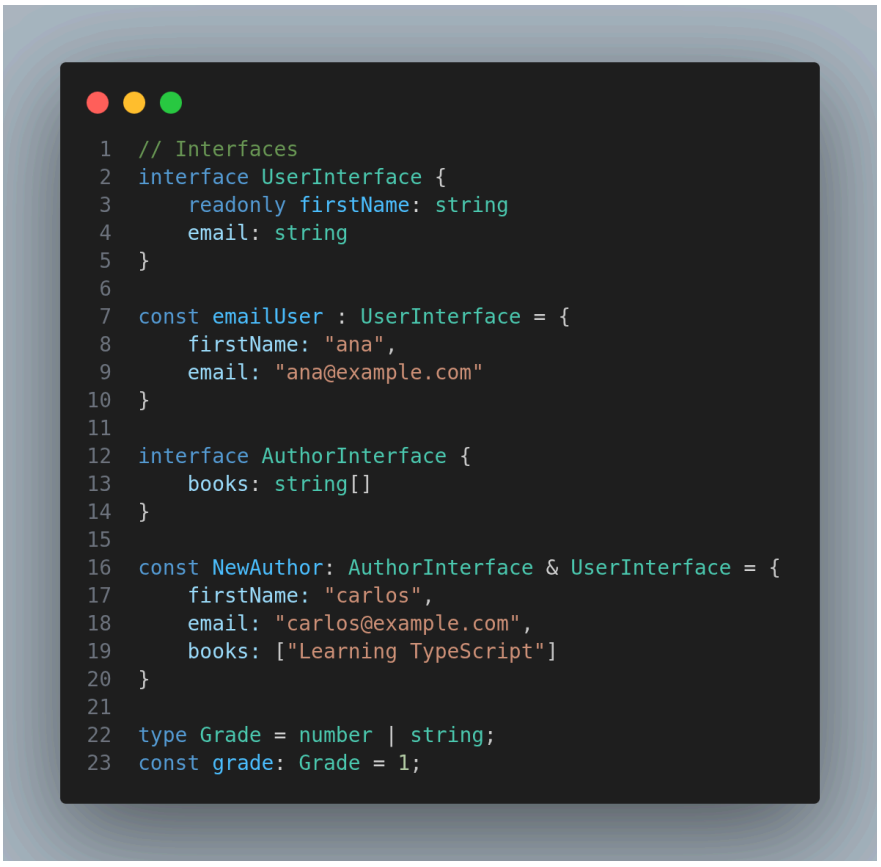
Union



```typescript
// Union
type Author = {
    books: string[]
}

const author: Author & User = {
    firstName: "maria",
    age: 45,
    email: "maria@example.com",
    orders: [],
    books: ["TypeScript Basics", "Advanced TypeScript"]
}
```

Aqui temos o exemplo de unions utilizado com objetos como nesse caso onde é criado o autor, mas o autor também é um usuário, então o mesmo tem os atributos tanto de autor quanto de usuário.

Interfaces



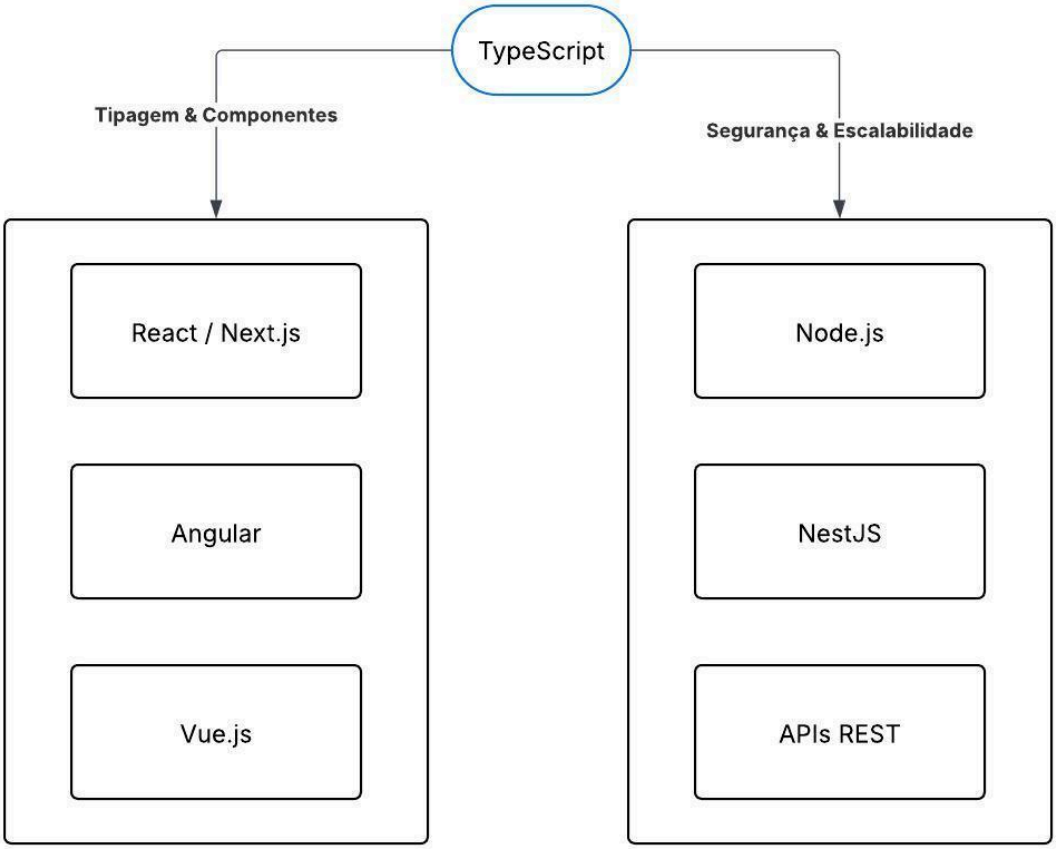
```typescript
// Interfaces
interface UserInterface {
    readonly firstName: string
    email: string
}

const emailUser : UserInterface = {
    firstName: "ana",
    email: "ana@example.com"
}

interface AuthorInterface {
    books: string[]
}

const NewAuthor: AuthorInterface & UserInterface = {
    firstName: "carlos",
    email: "carlos@example.com",
    books: ["Learning TypeScript"]
}

type Grade = number | string;
const grade: Grade = 1;
```

Bem parecidos com objetos e até mesmo citado pelo autor para você utilizar o que mais lhe agradar existem as interfaces que uma de suas diferenças é poder usar, por exemplo, o “readonly” que deixa um valor apenas para a leitura, e um exemplo de coisa que os objetos podem fazer e as interfaces não, está no último exemplo de definir um com mais de uma opção de tipagem como no exemplo ou um número, ou uma string.

No decorrer do vídeo também foi citado dois tópicos que não mudaram tanto do padrão, então nem fiz um ponto de exemplo deles como os anteriores, mas são basicamente as **classes** que não se diferem muito das classes normais de linguagem orientadas a objetos como C# ou Java, mas aqui no TS temos uma vantagem que podemos utilizar as classes como tipos. E o outro tópico são os **generics** uma ferramenta para criar componentes ou funções reutilizáveis, basicamente ao invés de utilizarmos uma tipagem fixa como string ou number utilizamos uma variável de tipo que permite que uma mesma função aceite diferentes tipos de dados.

Agora sobre o artigo "**Is TypeScript Frontend or Backend?**" ele reforça algumas coisas que vimos no vídeo anterior, as diferenças de tipagens estáticas e dinâmicas, mas agora nos aprofundamos mais tecnicamente sobre o TS como seu desempenho tanto no front-end quanto no back-end. Começando pelo front-end ele se mostra muito competente por sua compatibilidade quanto com o Angular e React que são algumas das principais ferramentas utilizadas no front-end que permite a construção de sites modernos e escaláveis, agora sobre o back-end o TS é uma das principais "linguagem" na construção de APIs graças a sua tipagem estática que aumenta a confiabilidade, reduz a carga cognitiva gasta para compreensão do código por outros programadores e assim facilitando ajustes, atualizações e futuras implementações.

Outro ponto importante citado neste artigo é a comparação entre o TS e o PHP onde nos é mostrado algumas vantagens e desvantagens de utilizar cada um deles onde a maior vantagem do TS é graças a sua tipagem estática onde entrega mais segurança e facilidade na compreensão do código, além disso, sendo mais atualizado e moderno, outro ponto abordado é a possibilidade de uma abordagem hibrida mesclando a implementação com o PHP e o TS assim utilizando um para contornar as limitações do outro.



Para elaboração do meu código próprio para teste das coisas passadas no decorrer deste card preferi utilizar mais as coisas que já tinham sido passadas e algumas coisas extras que precisei pesquisar para deixar o sistema funcional, desta vez por ainda não termos visto nada sobre desenvolvimentos de telas com Type Script preferi apenas estar fazendo um sistema simples de gerenciamento de uma biblioteca que roda no terminal sem interface gráfica, utilizei de auxílio de IA para configuração para rodar diretamente no terminal e me ajudar com algumas dificuldades que tive no desenvolvimento do projeto.

Mas basicamente é um sistema de biblioteca onde utilizei estruturas para criar o usuário e os livros e os autores e um sistema de funções para aluguel do livro, com verificações se o livro esta ou não disponível e o cálculo de se o aluguel for efetivado para quando devolver os livros.

Os códigos elaborados no decorrer do card. estão todos contidos no seguinte repositório:

https://github.com/SebenJose/Fast-Camp---Web/tree/main/Card%203%20-%20Pr%C3%A1tica%3A%20Iniciando%20com%20TypeScript%20(II)

Onde Códigos Aula são os códigos feitos acompanhando a videoaula e código pessoal são os códigos realizados para prática.

Dificuldades

Algumas dificuldades que enfrentei no decorrer deste card. foram primeiro na videoaula algumas coisas da explicação dele eu não havia entendido de primeira então para tirar essas dúvidas que eu havia ficado consultei a IA mostrando o código que eu não havia entendido e explicando minha dúvida assim com um pouco de “conversa” consegui tirar essas dúvidas e entender melhor, e a outra dificuldade foi na elaboração do código para prática onde nesse por não ter muita familiaridade com o Type Script tive alguns probleminhas de sintaxe e algumas coisinhas, não sabia como fazer especificamente nesta linguagem, mas com algumas pesquisas logo consegui resolver esse problema.

Conclusões

Com esse card já conseguimos criar uma boa base sobre o Type Script e para pessoas como eu que nunca haviam trabalhado com o mesmo ter uma primeira impressão já colocando a mão na massa e aprendendo na prática seu funcionamento e para oque aplicar.

Referencias

Vídeo:

Curso de TypeScript para Completos Iniciantes

Artigo:

https://accesto.com/blog/typescript-front-or-back-end/