# Typescript

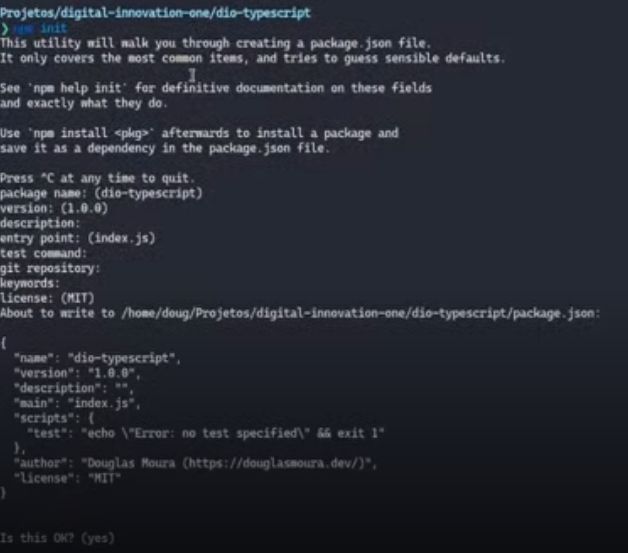
## Instalação do typescript

- instalar o typescript no computador <https://www.typescriptlang.org/pt/>

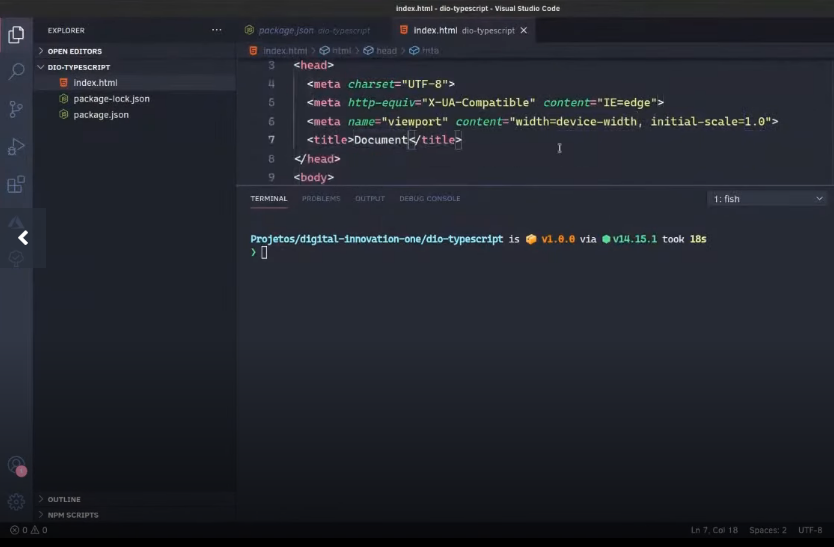
- iniciar o git bash ou o typescript

- colocar o comando no terminal “npm init”

- vai aparecer essa janela para realizar as configurações



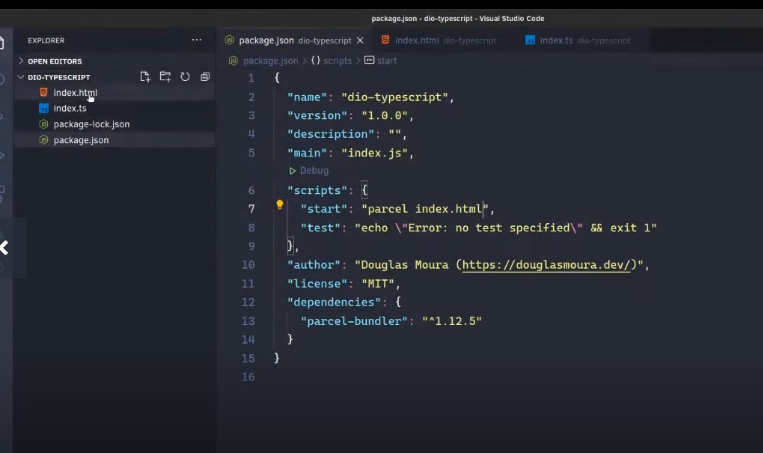
- a extensão também aparecerá no VS Code



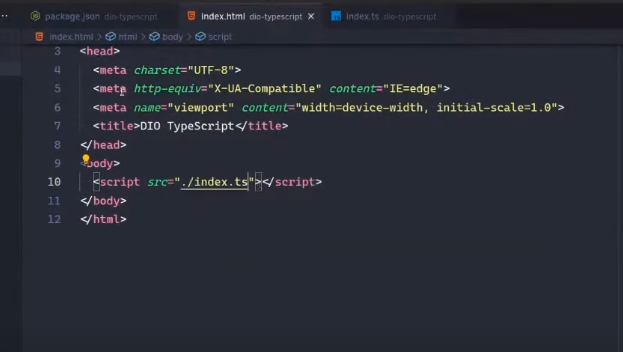
- criar arquivo index.ts (esse será o arquivo typescript)

- esses são os arquivos que irão existir dentro da pasta

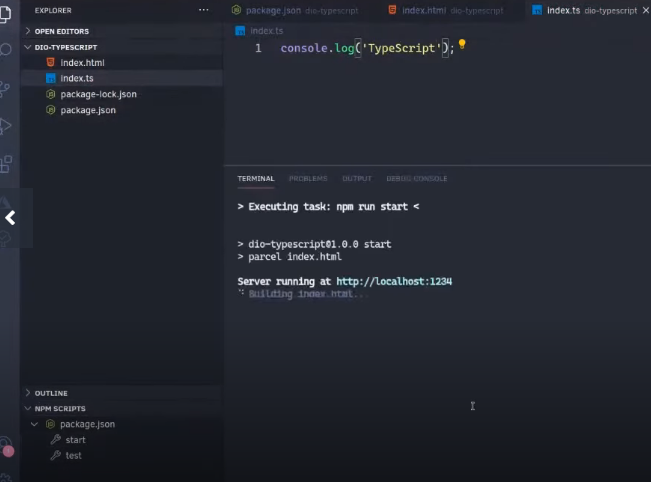
- o que precisa estar dentro do arquivo package.json



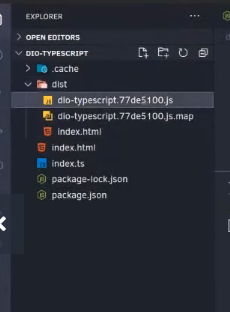
- isso é o que vai conter dentro do index.html



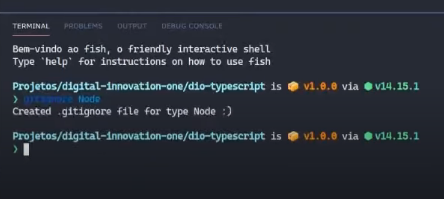
- abrir o arquivo index.ts e digitar o seguinte código e clicar o start



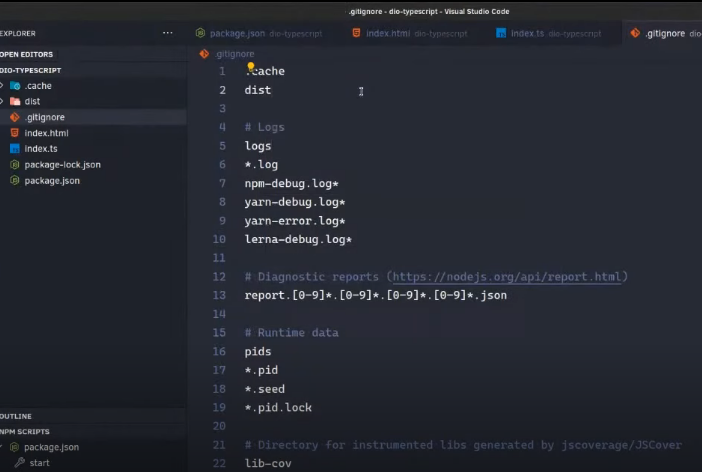
- será acrescentado vários arquivos de cache na pasta



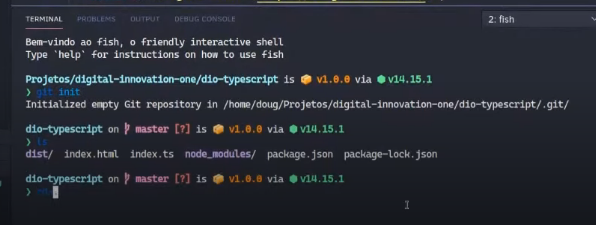
- digitar o comando gitignore node



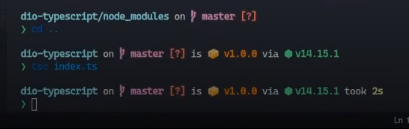
- vai aparecer na pasta o arquivo .gitignore, abrir e digitar os seguintes comandos dentro desse arquivo .cache e dist



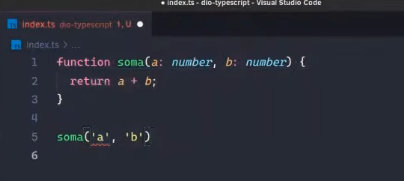
- dar o comando git init no terminal do VS Code



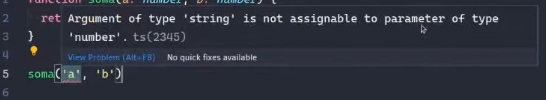
- quando temos acesso ao typescript compiler, e o arquivo pode ser chamado direto no terminal do VS Code digitando tsc index.ts



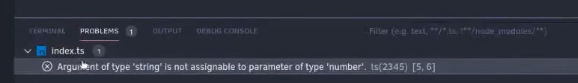
- exemplo de coisas que o typescript faz quando estamos desenvolvendo uma função por exemplo



- se colocar strings na soma com letras dentro, já aparecerá uma janela de erro em cima da string quando você passa o mouse

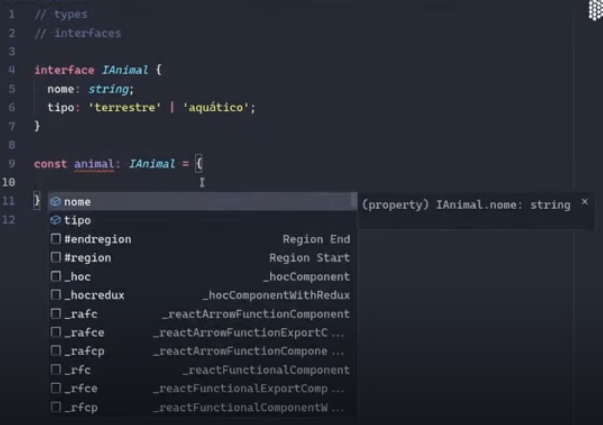


- e no VS Code aparece o erro na aba problems do terminal

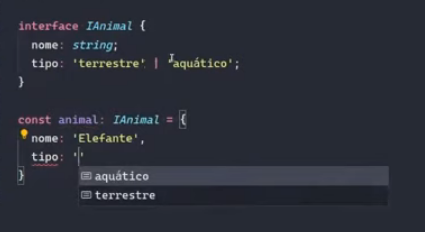


## Como funcionam as interfaces

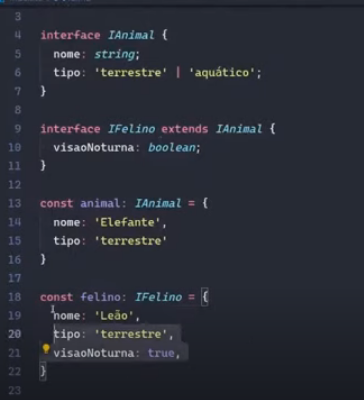
-quando você digita ctrl + barra já irá aparecer os tipos que eu defini dentro de interface



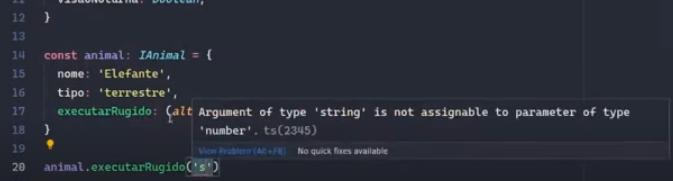
- quando você começa a digitar já começam a aparecer os tipos que foram definidos anteriormente, e precisa necessariamente ser um dos tipos definidos. Não pode acrescentar outro



- essas interfaces poderão ser estendidas conforme necessidade



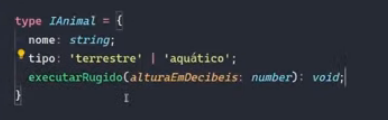




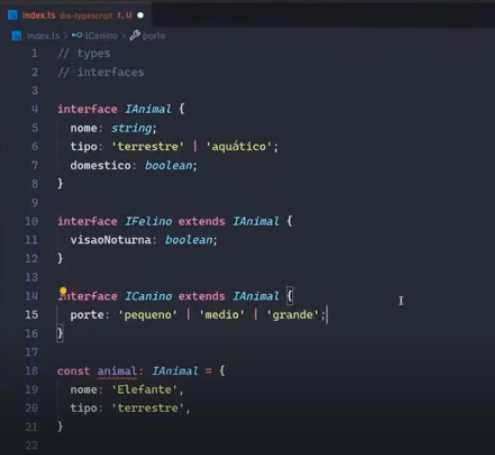
- as interfaces são usadas para definir contratos de uma classe que vamos definir

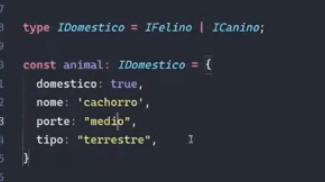
## Como funcionam os types

- o uso do type é bem parecido com o uso da interface no TS, só que na frente do nome criado precisa do =



- o type nós usamos para juntar diferentes interfaces





- podemos definir o type de acordo com os itens dentro da interface

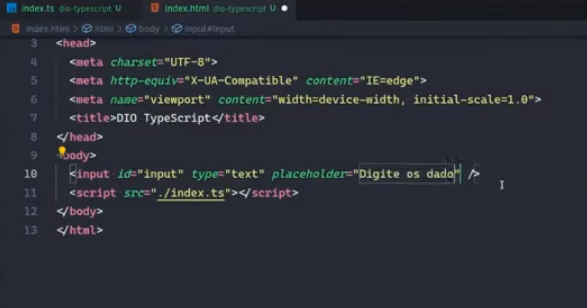
- se for necessário pode ser trocado | pelo & caso você queira juntar duas classes diferentes, porém será necessário atribuir valores para todos os itens, se não dará erro de que falta algum valor a ser atribuído.

## Tratando a TAG Input

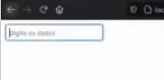
- por enquanto estamos somente fazendo os códigos, não temos uma interface definida

- o input serve para criar um local de armazenamento de dados que serão inseridos, geralmente com um placeholder indicando o que deverá ser escrito no campo em questão

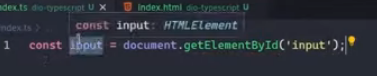
- exemplo da estrutura do HTML



- o input com placeholder ficará assim no HTML



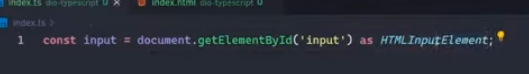
- dentro do index.ts começamos a digitar como deverá ser o código



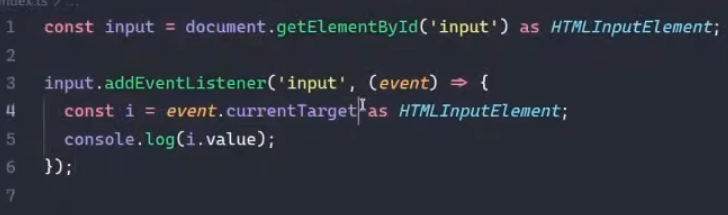
- quando você passa o mouse em cima do ‘input’ ele identifica o tipo do elemento que estamos utilizando

- <https://developer.mozilla.org/pt-BR/docs/Web/API/HTMLElement>

- agora vamos fazer esse tipo de elemento genérico da pagina se tornar um valor atribuído e verdadeiro



- com esse as HTMLInputElement, nos mostramos para o código que o input terá um valor

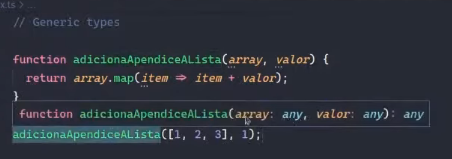


- não pode colocar o comando antes com : , pois o elemento esta sendo definido após o =, portanto o HTMLInputElement antes fica como null

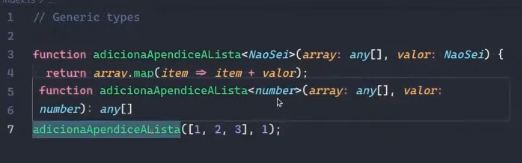


## O que são generic types

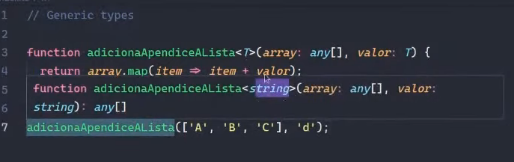
- como não estou definindo o que a minha lista recebe e nem o que ela retorna, os valores no TS serão definidos como any



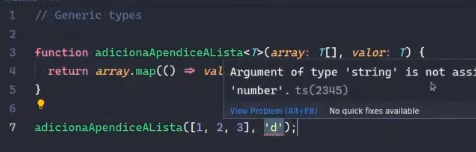
- como tipar os dados, nesse exemplo abaixo definimos o adicionaApendiceALista como <NaoSei>, o array como um any[] e o valor também como NaoSei. E o TS retornou como serão identificados dos valores



- o TS identificou os valores de NaoSei como numbers, geralmente quando não sabemos do que se trata a lista que estamos criando colocamos a letra T, por convenção. Como o T não é um valor definido, ele infere os valores de acordo com os informados, ou seja, se eu coloco número, ele identifica números, se eu coloco textos, ele identifica strings

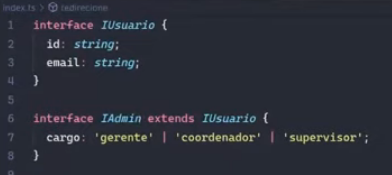


- porém se eu definir um valor para T nos valores informados, todos eles deverão ser do mesmo tipo, ou tudo number, ou tudo string, etc. Pois se eu definir um texto e um valor para os valores de T, acontecerá um erro:

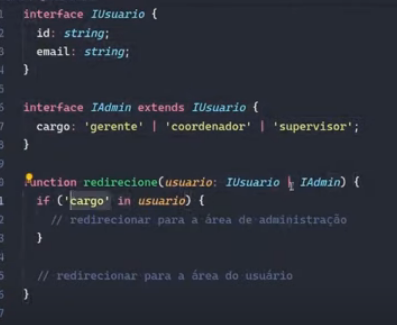


## Desenvolvendo condicionais a partir de parâmetros

- iniciamos criando/ definindo uma interface

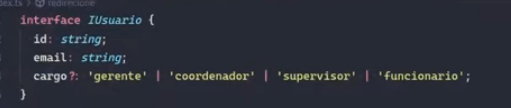


- fazer uma função de redirecionar e utilizar o in pra pesquisar um item dentro de outro



## Utilizando o caracter “?” para variáveis opcionais

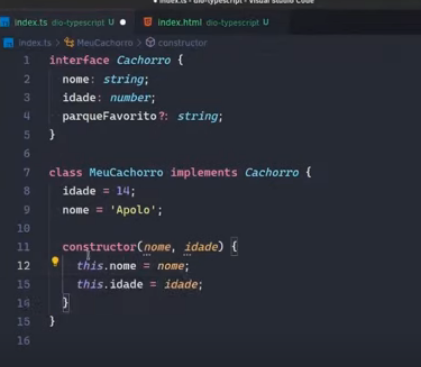
- se quiser colocar um item como opcional no preenchimento, podemos recorrer ao caracter “?” na frente do :

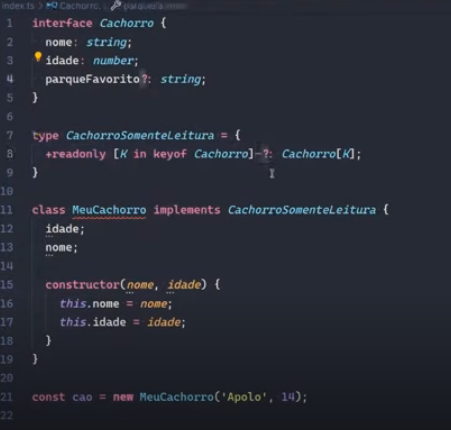


- pode-se utilizar o if para chamar esse dado



## Criando variáveis com propriedade readolny e private

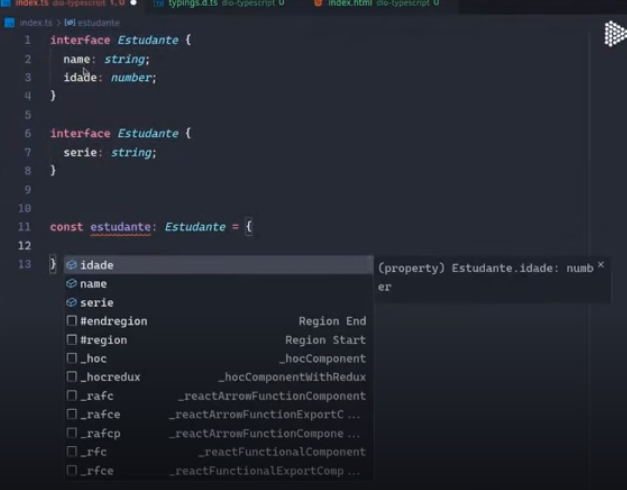




- coloca-se o + na frente do readonly apenas por melhoramento da leitura do código para quem for utilizar seu código e o -? para indicar a retirada do item

## Como importar bibliotecas com typescript

- quando se cria varias interfaces dentro do código com a mesma nomenclatura e com itens diferentes, na constante eles terão seus valores de itens somados, conforme exemplo abaixo



- cria-se um arquivo chamado de typings.d.ts



- dentro do index.ts coloca-se para fazer o import da biblioteca externa por exemplo jquery, quando se cria uma nova função, ela não entra automaticamente, ela entra apenas como um texto no VS Code. Abc – indica apenas texto. Ou seja, ele não está pegando o método do jquery.

- para fazer funcionar o jquery é necessário

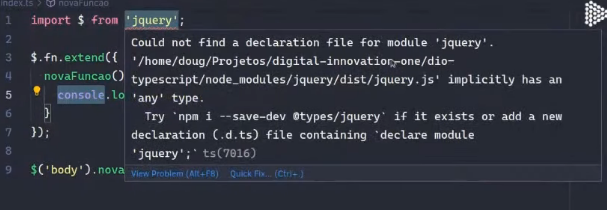
- dar o comando tsc –init



- ele vai criar o tsconfig.json

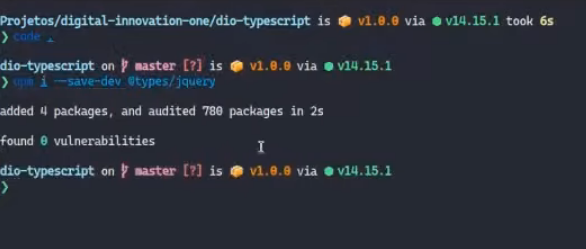


- ele vai mostar o erro no TS



- para resolver esse problema precisa ir no terminal e instalar o jquery com o seguinte comando:

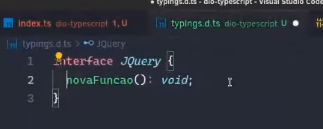
npm i –save-dev @types/jquery



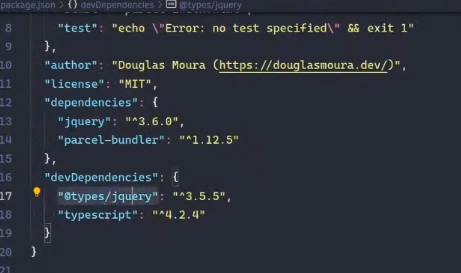
- como ele sabe as funções existentes dentro do jquery ele dará um erro caso você crie uma nova função inexistente no jquery



- é necessário criar essa função dentro do arquivos typings



- ele cria dentro do arquivo package.json



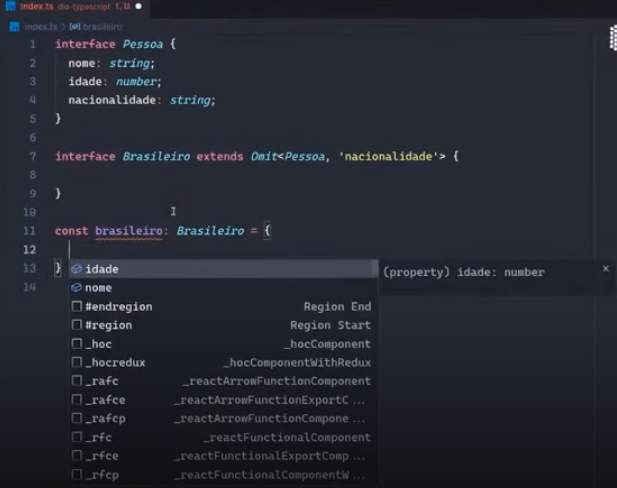
<https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>

## Exemplo de como usar Omit

<https://www.typescriptlang.org/docs/>

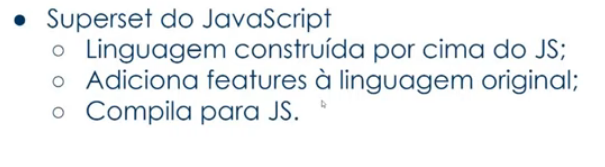
- dentro do arquivo index.ts quando você criar a interface e for necessário criar outra interface com algum item que já existe, usa-se o omit para ignorá-lo. Abaixo no exemplo, temos dentro da interface pessoa, o item a ser preenchido de nome nacionalidade. Porém estamos criando outra interface com o título de brasileiro (que já é uma nacionalidade).

Para não haver duas informações iguais repetidamente, usa-se o omit para quando esse item for aplicado, ser ignorado seu preenchimento. Pode-se observar que quando você começa a preencher os itens, a nacionalidade não aparece para ser preenchida, quando a interface brasileiro esta sendo utilizada.

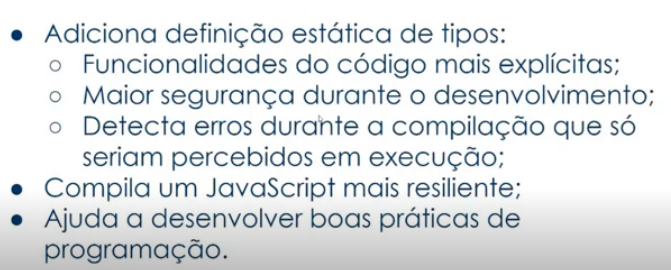


<https://douglasmoura.dev/>

## O que é TypeScript?



## Por que usar TypeScript?



## Criando uma aplicação simples de exemplo com JS

- realizar a instalação do typescript e do lite server

- fazer um html simples com dois inputs e um button