lucas indicou para instalar template de projeto react com typescript

<https://create-react-app.dev/docs/adding-typescript/>

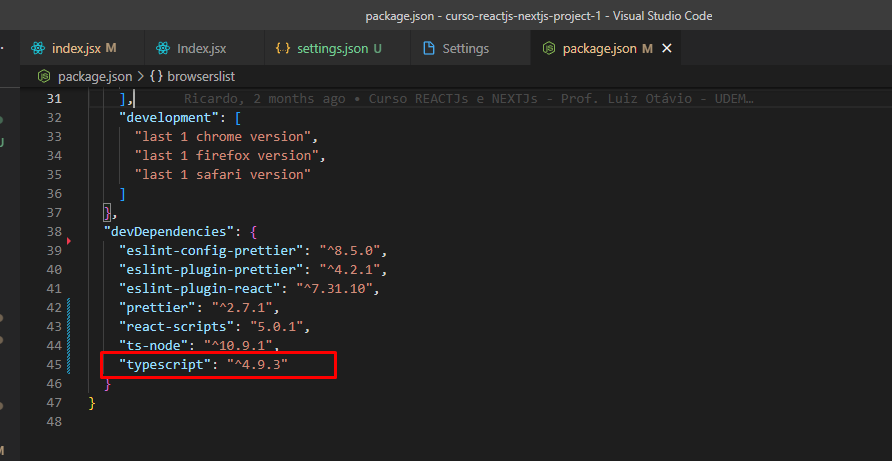
<https://nextjs.org/docs/getting-started>

1)Instalar o Typescript

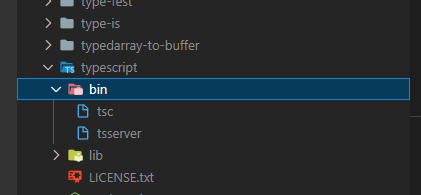
Vamos instalar o typescript por projeto, ou seja, não vai ser instalado de maneira global

npm i typescript –D

Podemos verificar no package.json nas devDependencies



- Depois de instalado, dentro da node modules (node\_modules\typescript), temos a pasta typescript com as pastas tsc e tsserver dentro :



Vamos mover estes arquivos para a .bin para utilizarmos o npx e poder executar os binários dos arquivos.

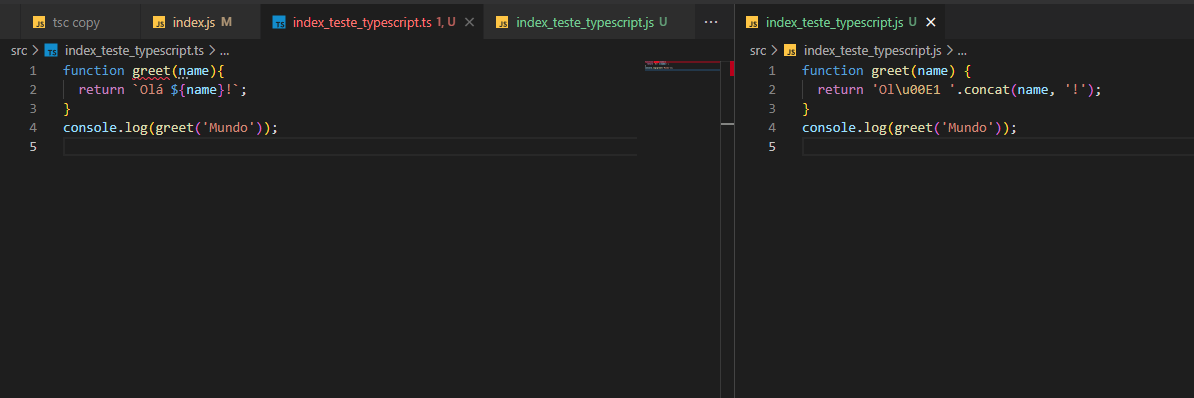
Para testarmos, vamos criar um arquivo chamado index.ts (typescript).

Ao rodar o comando de compilar, é gerado outro arquivo com o mesmo nome porém com extensão js.

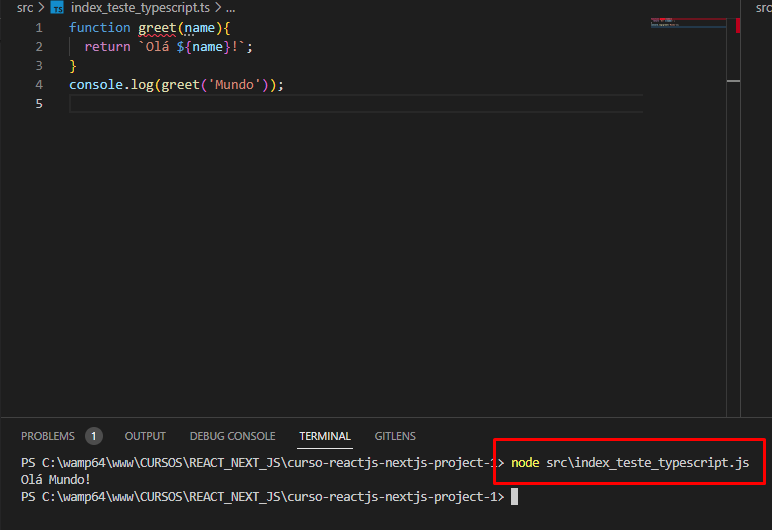
Então, como eu já tenho um index.js, eu criei um src\index\_teste\_typescript.ts.

Para compilar, vamos usar o comando npx tsc src\index\_teste\_typescript.ts

O sistema vai criar dois arquivos iguais, porem o da direita vai ser similar ao outro:

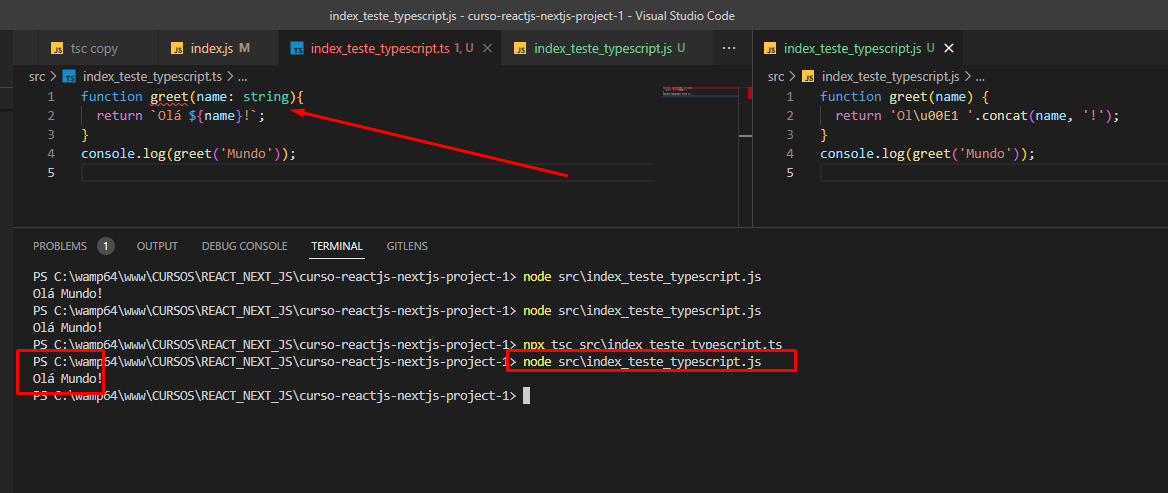


Com o comando do node + nome do arquivo, poderiamos testar um js normal



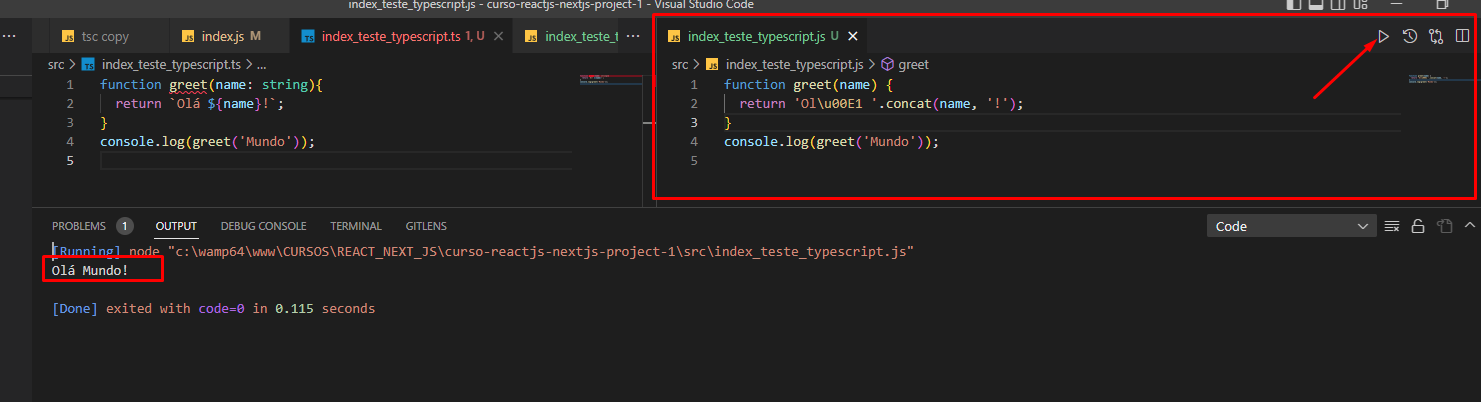
Porém, com o typescript, só compilando antes.

\*\* Na tela do curso, ele não conseguiu com o comando do node, porém no local eu consegui mesmo com a sintaxe de typescript, talvez devido a versão:



\*\* No local, ao clicar no botão de play, deu pau e no da aula

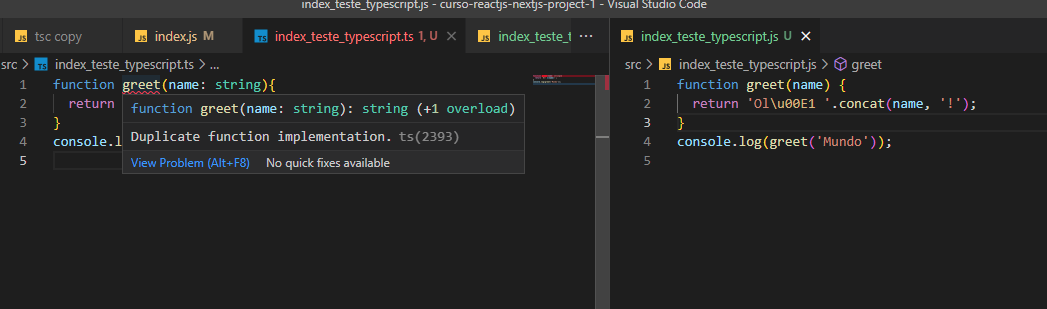
Já ao clicar no play do arquivo de produção (compilado), funcionou, então não tenho certeza se é um bug:



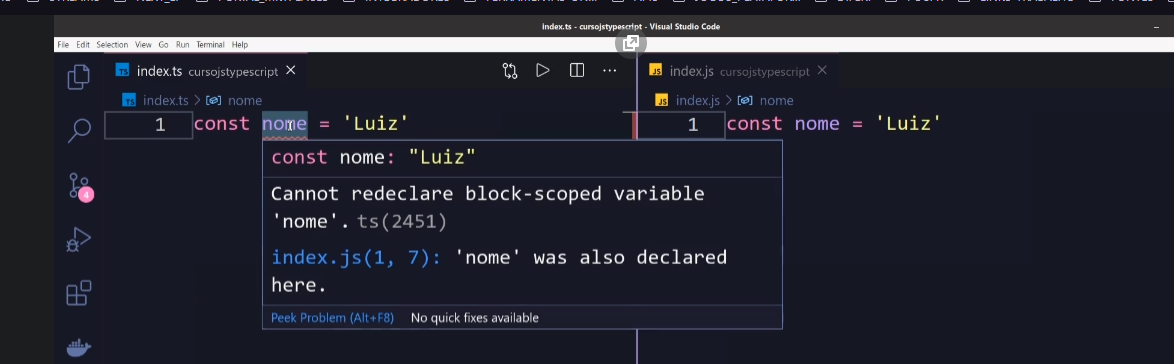
- O typescript tem dois modos de operação “module mode” e o “script mode” que funcionam de maneiras diferente.

No caso acima estamos utilizamos o scipt mode, que usaria um arquivo js enorme.

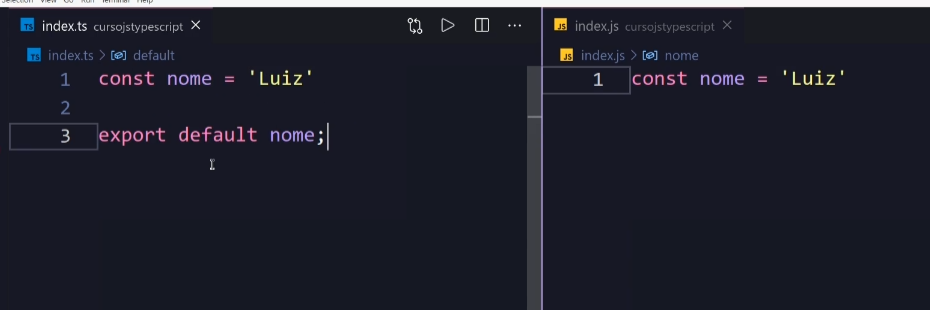
Por isso que ele interpreta que a função greet já existe, mesmo com dois arquivos separados:



No caso de uma constant, por exemplo, o script já reclama que não podemos redeclarar uma contante da mesma maneira como se fosse declarada a mesma variável duas vezes no mesmo script:



Para sairmos deste modo, precisamos apenas exportar ou importar alguma coisa, que ele já reconhece que saímos do script mode e assume que já estamos trabalhando no module mode

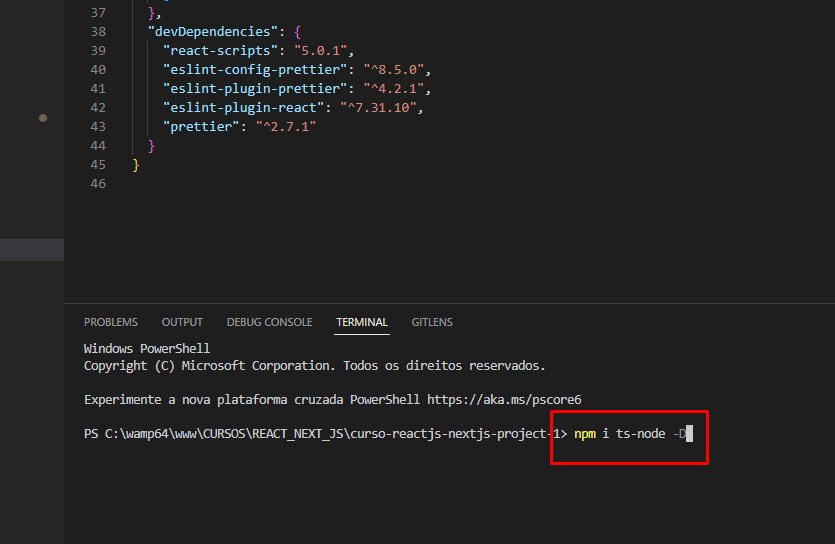


Então poderemos ver um export ou import apenas para utilizarmos nomes já utilizarmos em algum outro lugar.

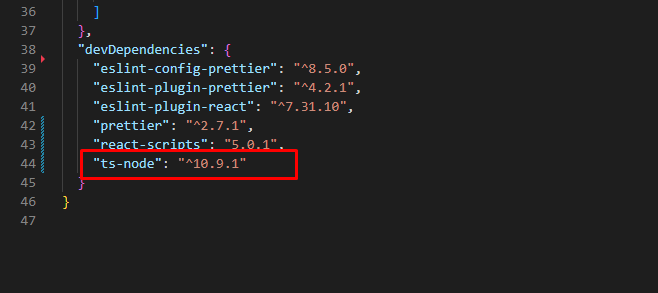
O javascript é uma linguagem interpretada e podemos ver os resultados dos códigos sem compilar porém, o typescript é compilado e para vermos nossos códigos rodando, precisamos utilizar uma ferramenta no VScode para testarmos mais rapidamente sem usar comandos no terminal.

2) Vamos instalar uma dependência de desenvolvimento chamada ts-node

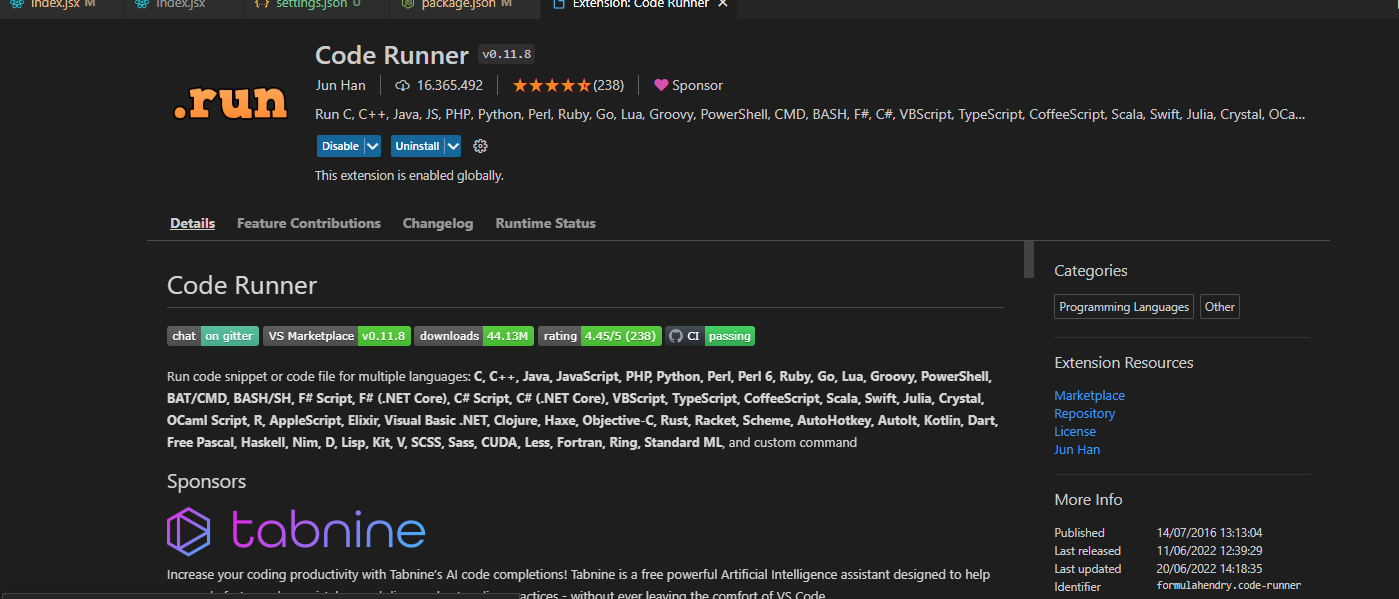
No terminal, vamos usar o npm i ts-node -D



O arquivo já vai estar nas nossas devDependencies no package.json



Isso já vai habilitar o code runner que é uma extensão do vs code e vamos utilizar para executar diretamente no terminal, usando o botão de play do vscode.



Após instalar o code runner, precisamos configurar e para isso precisamos criar uma pasta na raiz do projeto com o nome .vscode (com o ponto) e vamos utilizar para configuração

Dentro vamos ter um arquivo settings.json que será uma configuração local do para o vscode que não seria global, ou seja, não vai alterar outros projetos.

.vscode\settings.json

Um exemplo seria o zoomLevel:



Porem vamos inserir o execution map, retirando as partes que não precisamos, vai ficar assim

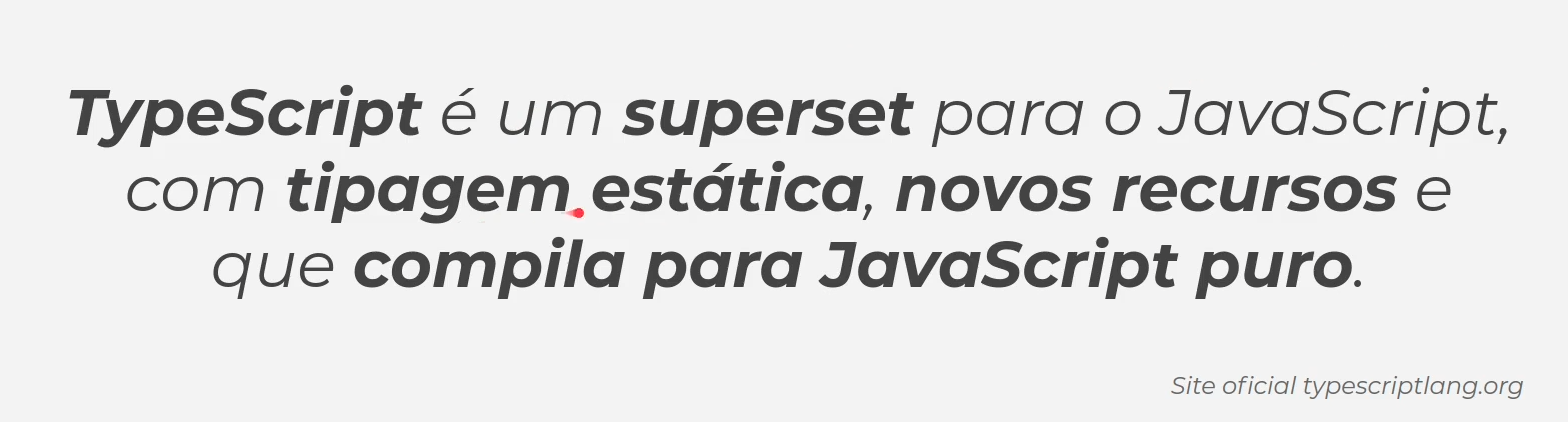
{

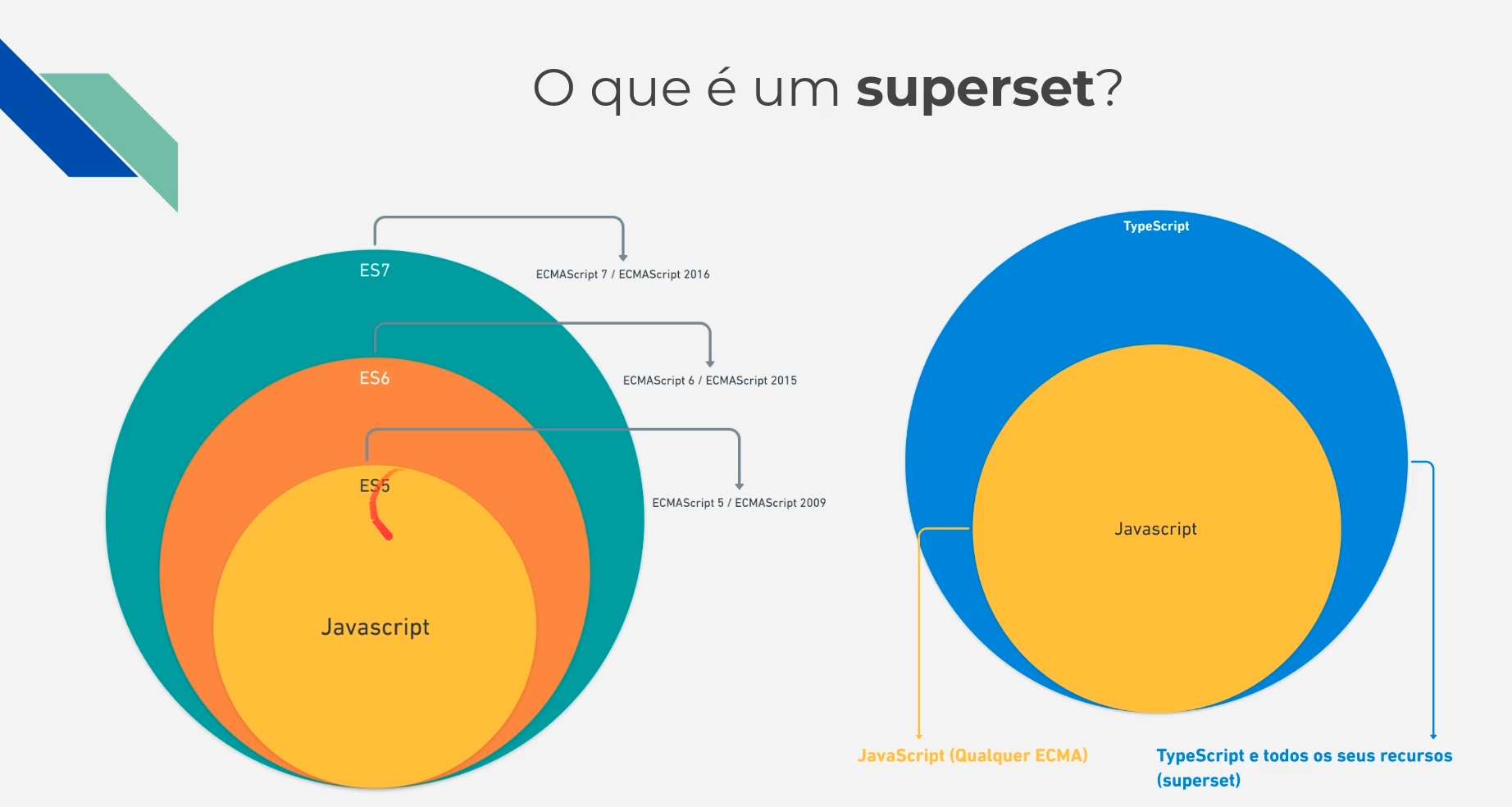
  "code-runner.executorMap": {

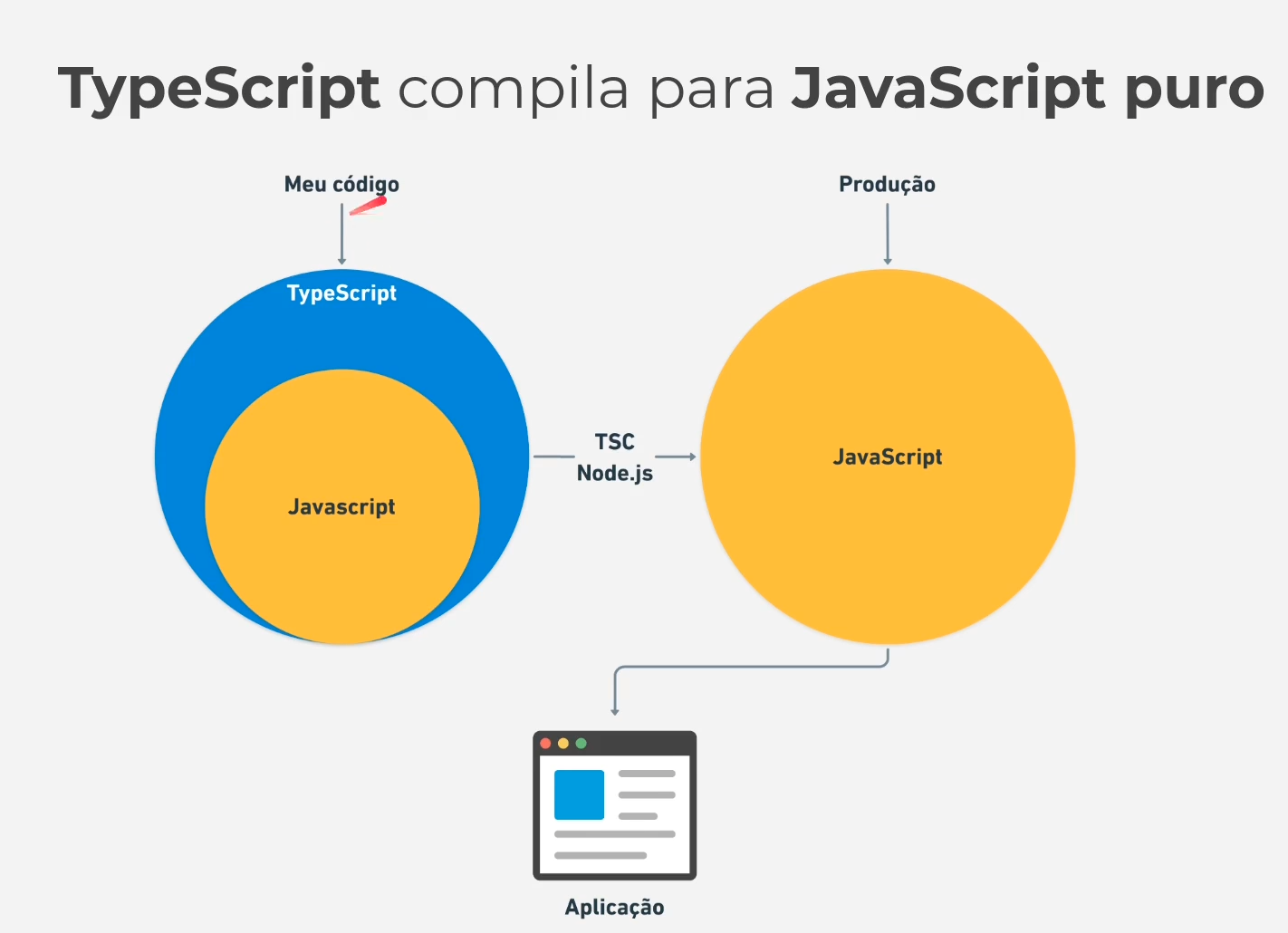
    "typescript": "ts-node",

  }

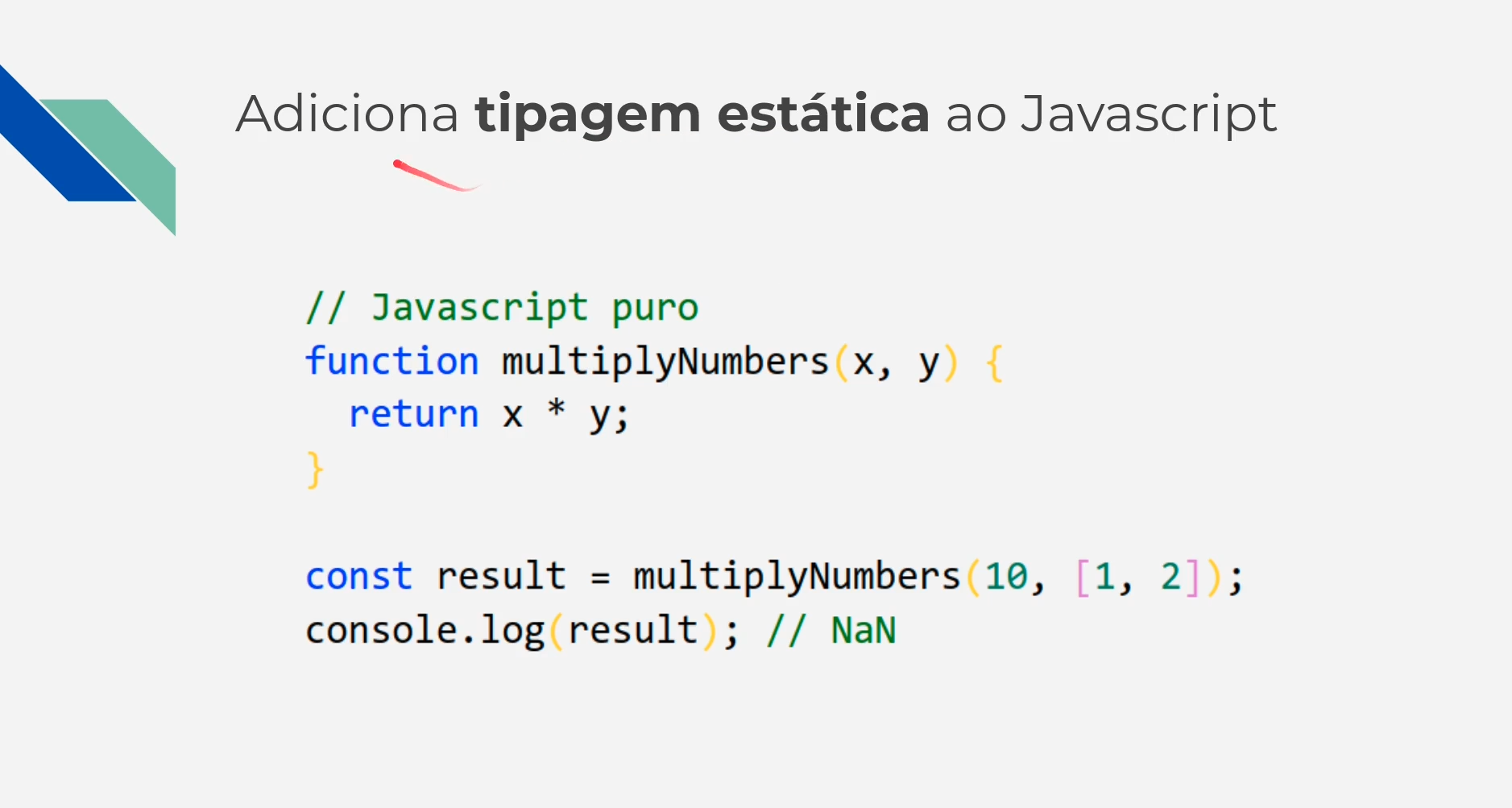
}



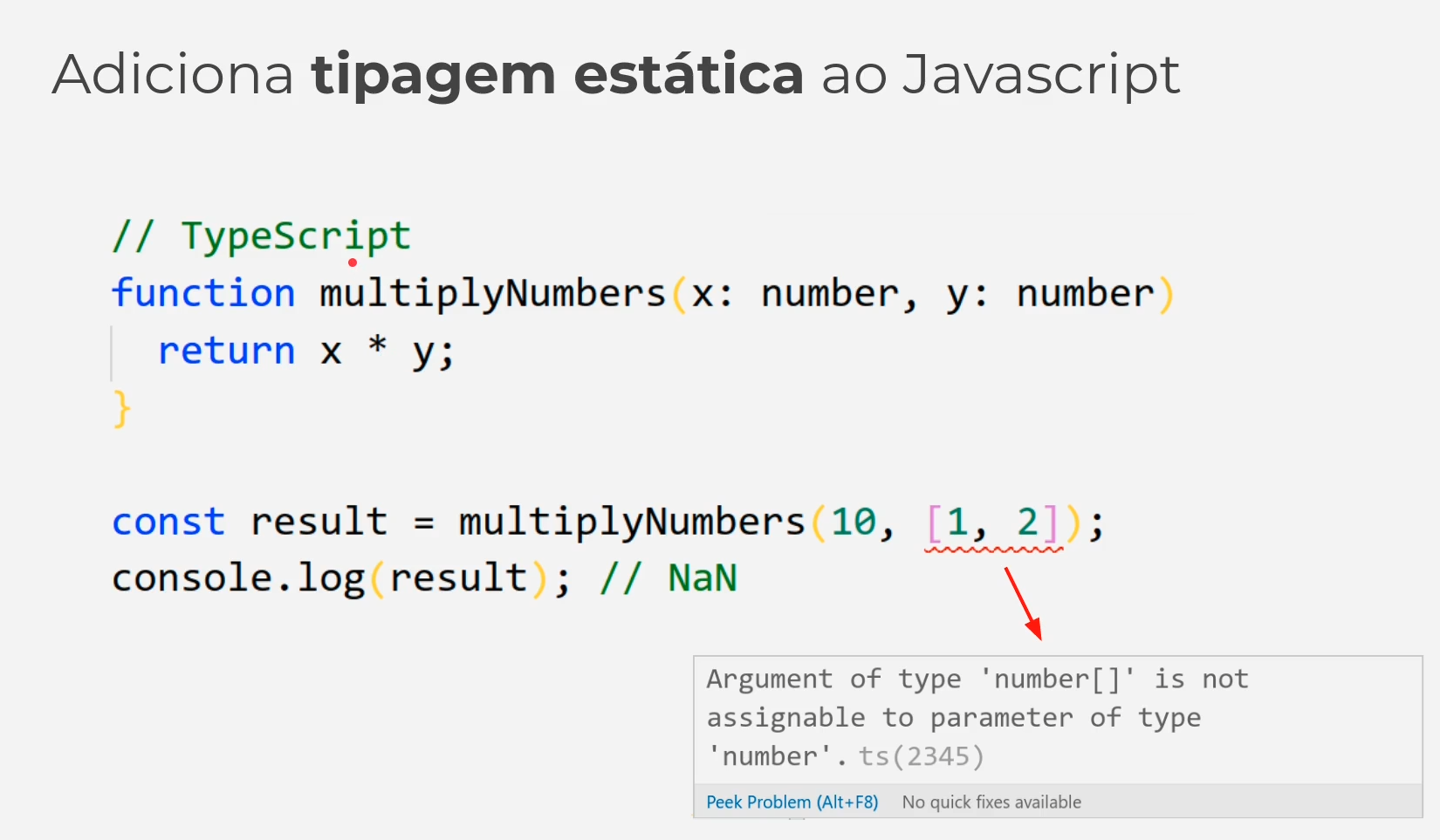


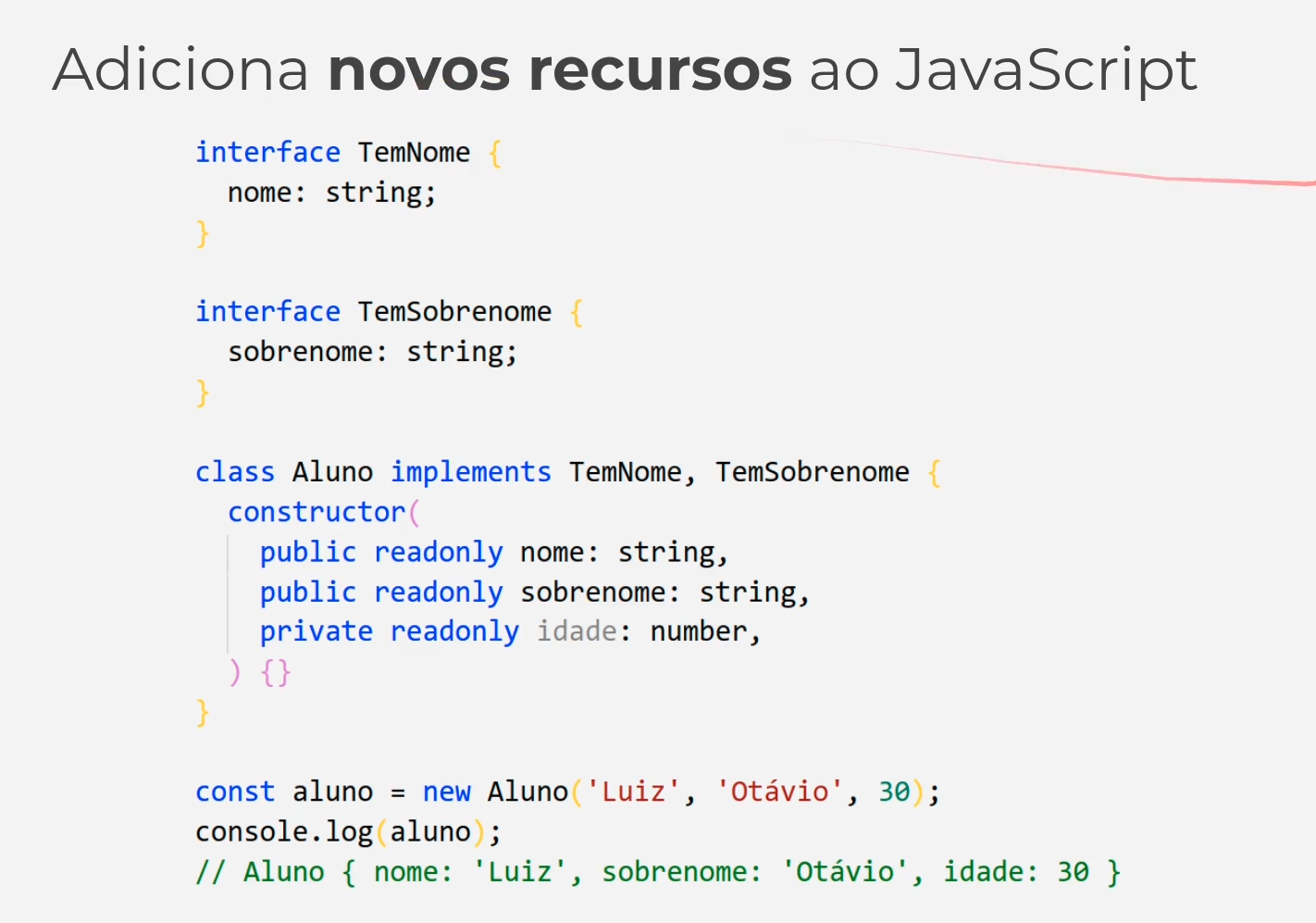


No javascript o erro só vai ser acusado no runtime (quando estiver executando).

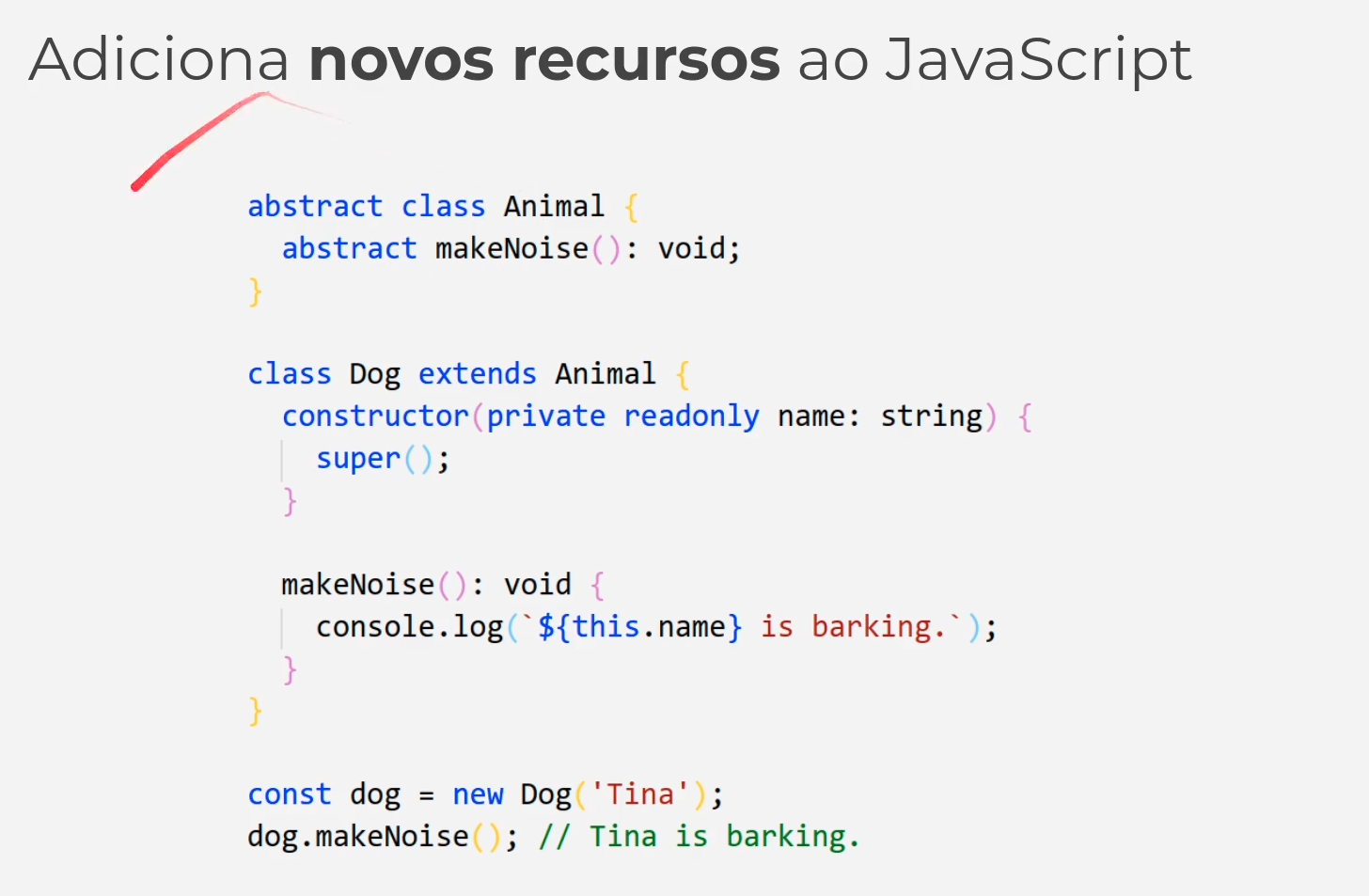


Com o typescript os tipos já são informados, evitando os erros anteriores:

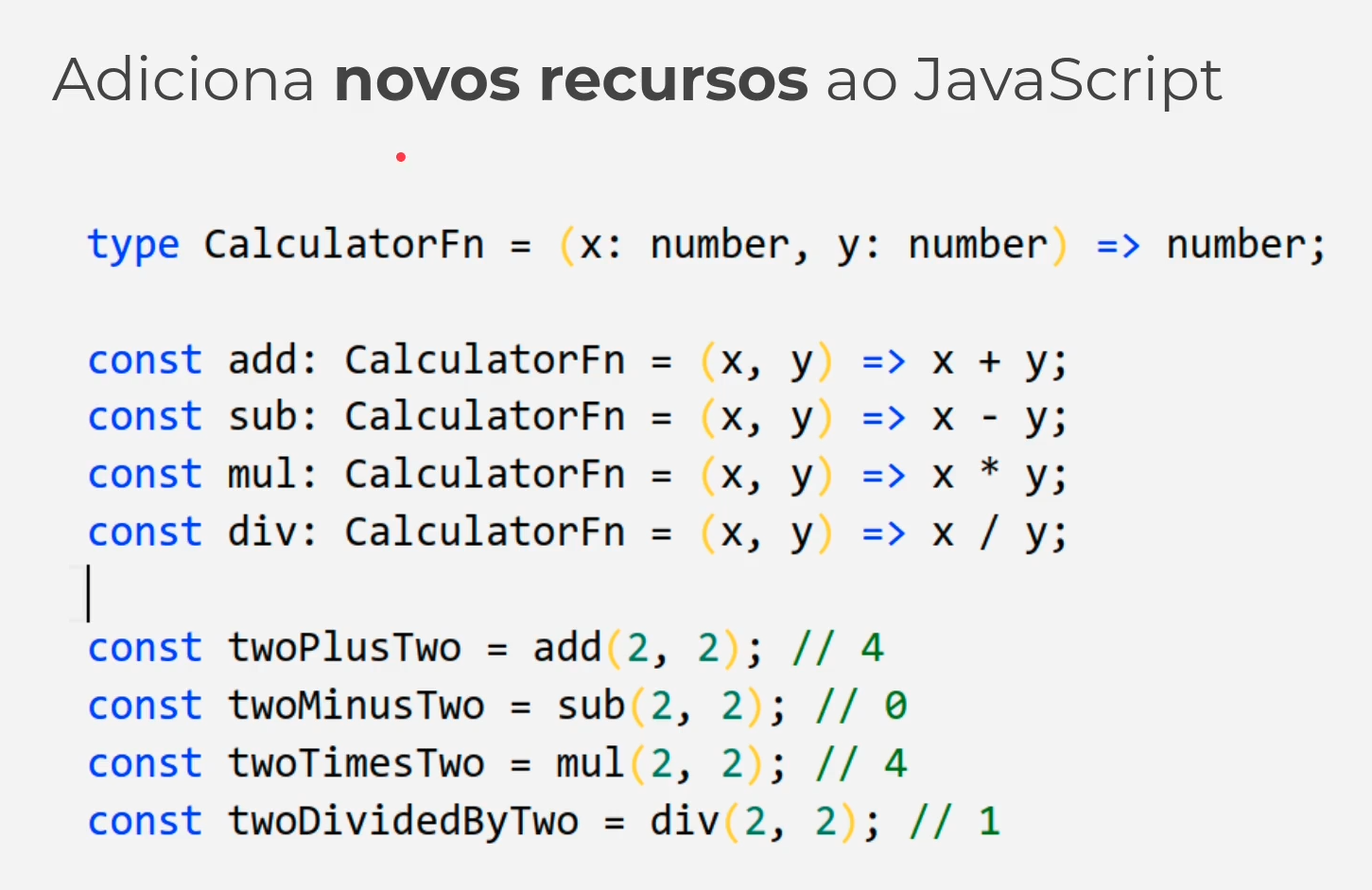


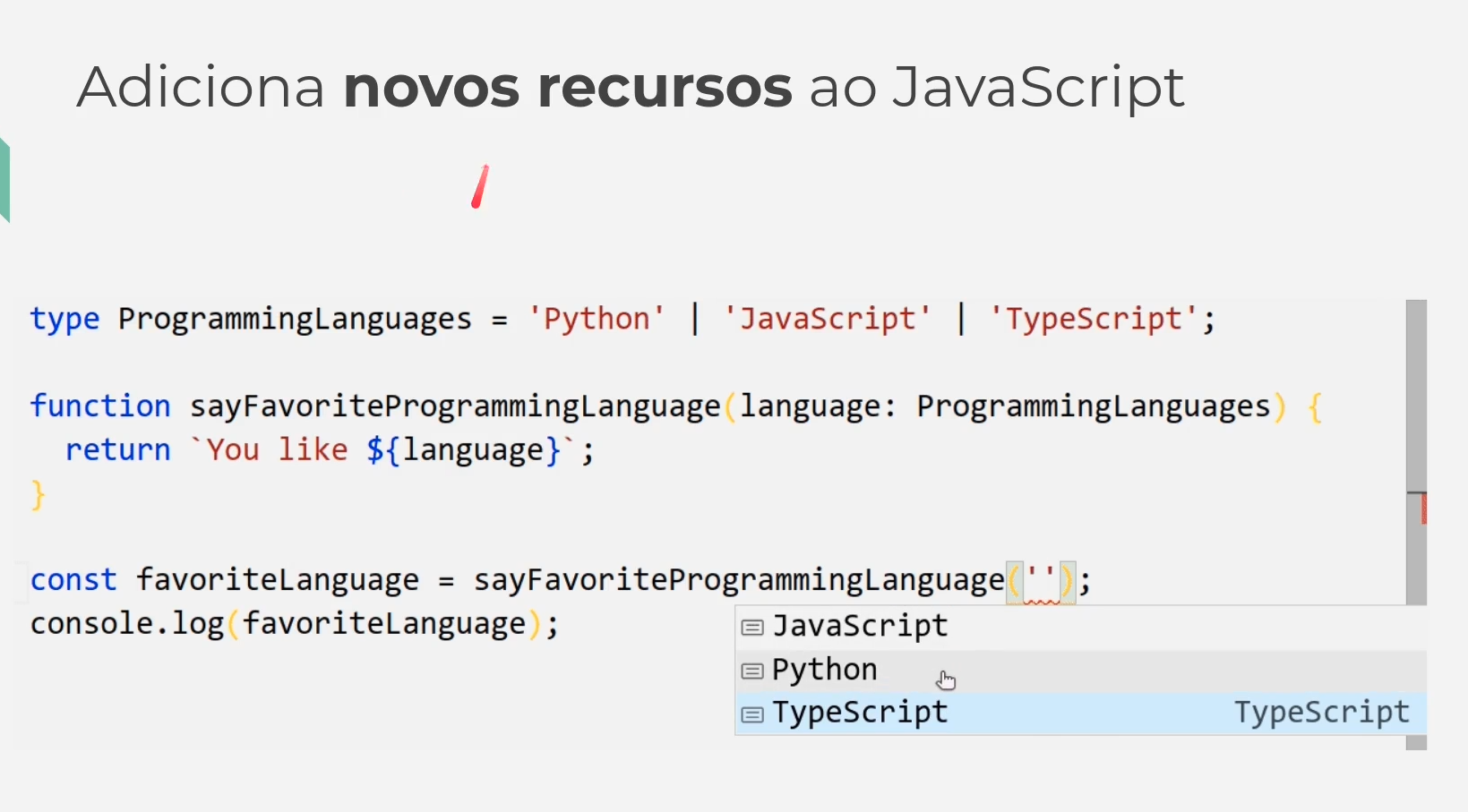


Ex. com abstratos:



\*Na seta é o retorno



 Multipla escolha (acima tem uma espécie de enum) definimos tipos e só poderemos mandar valores correspondentes.

autocompletar:



Abaixo, os tipos são inferidos automaticamente:

