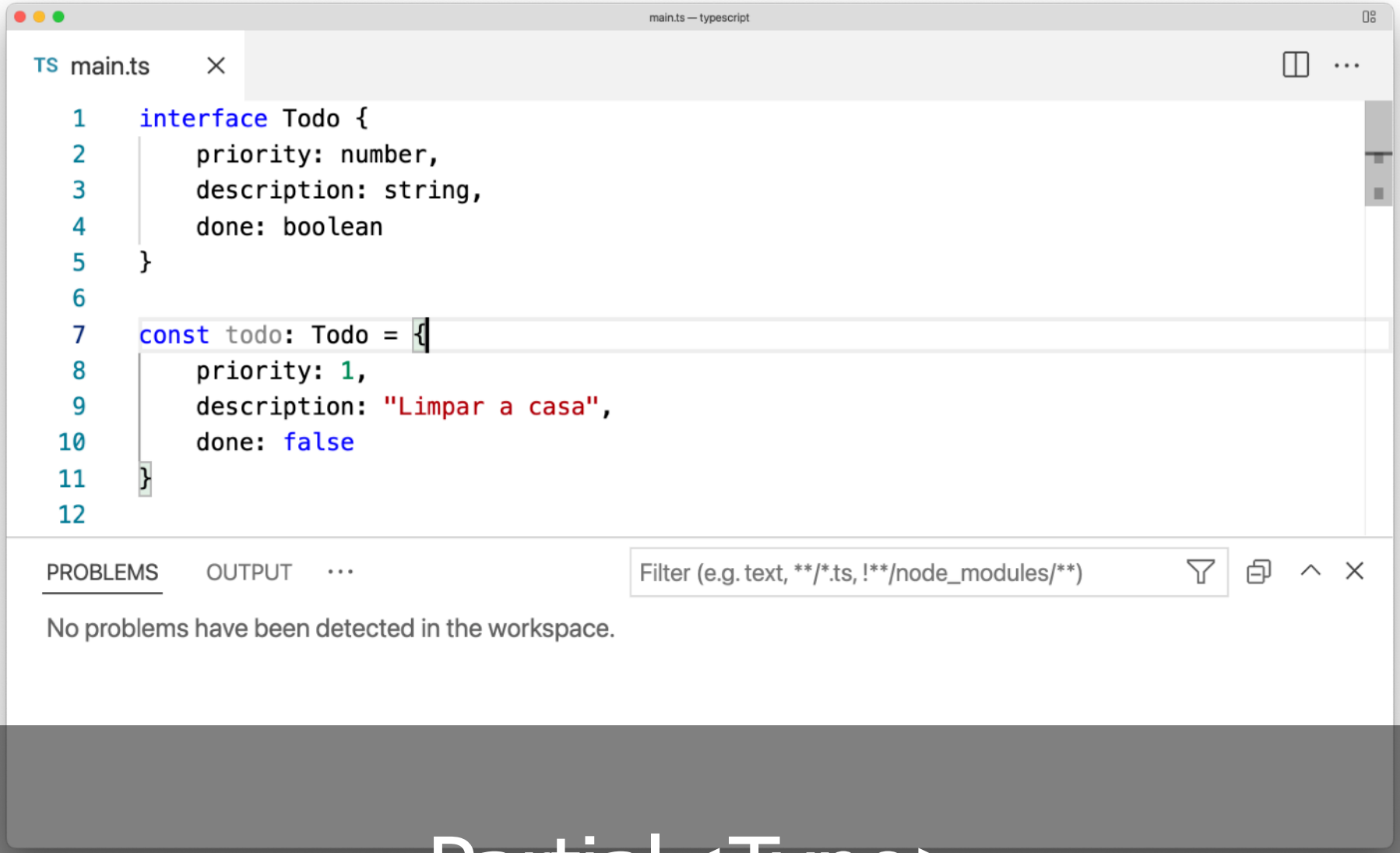
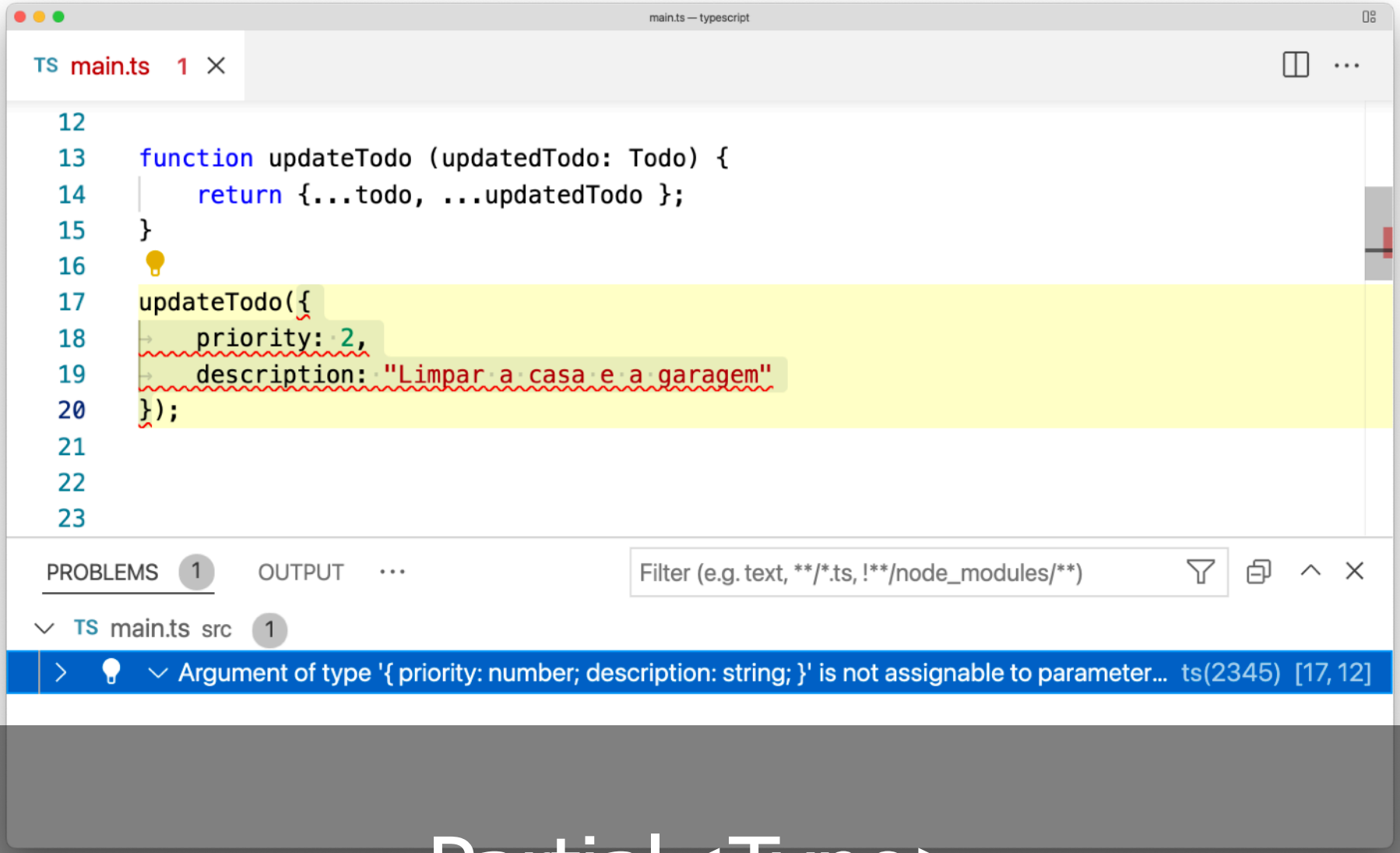

TS
TypeScript

Utility Types

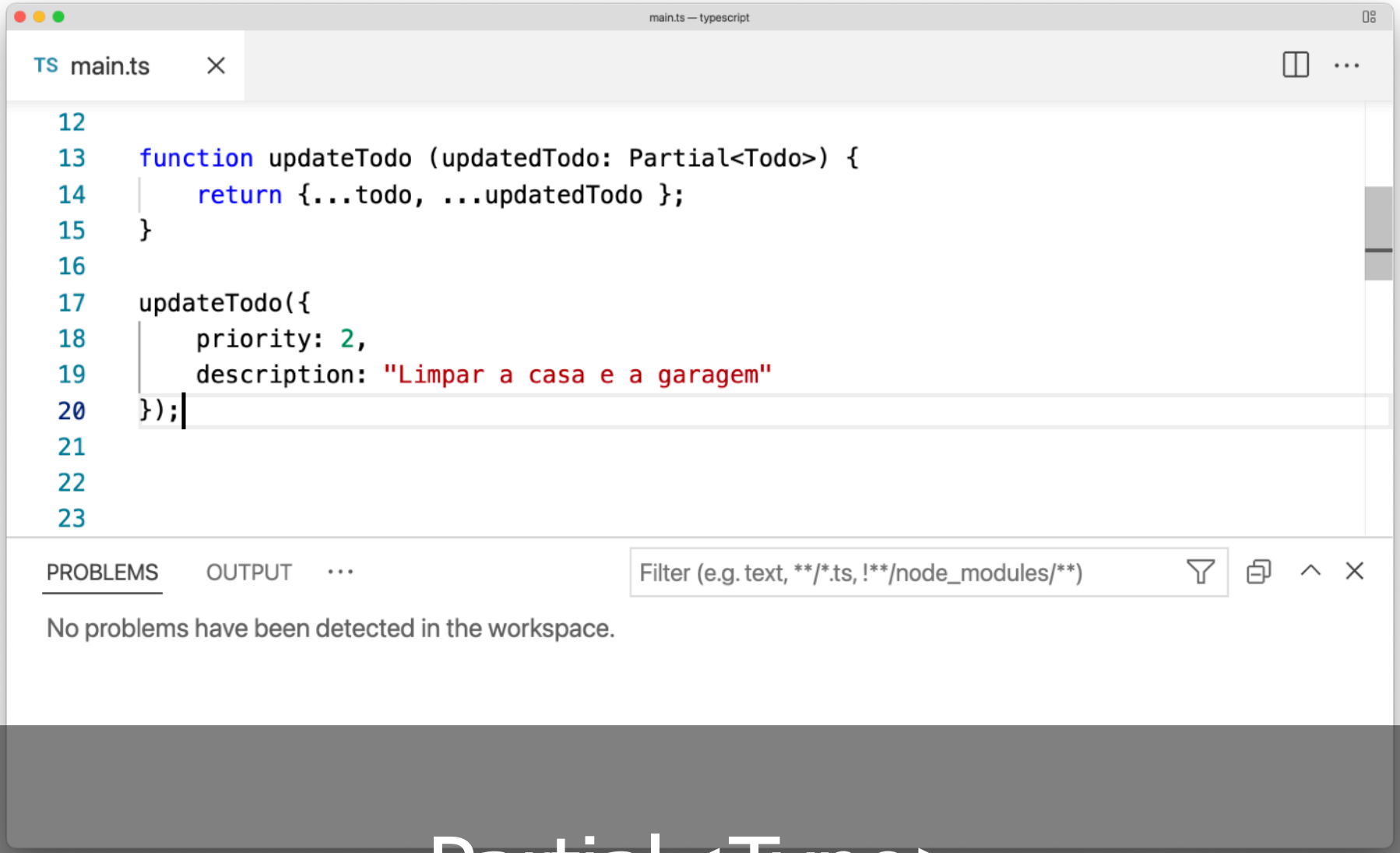
O TypeScript suporta o novo sistema de módulos criado no ES6 por meio das palavras-chave import e export



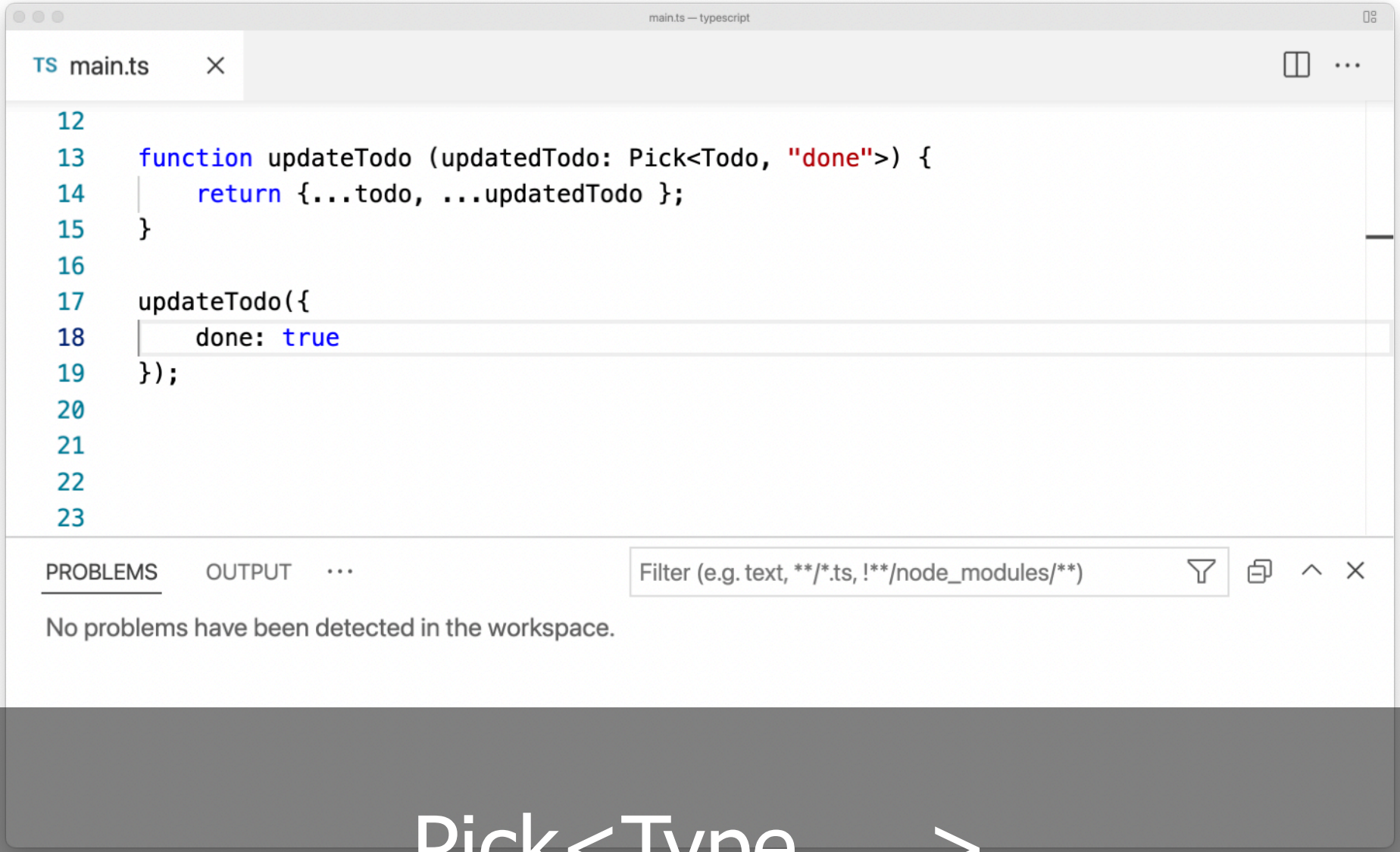
Partial<Type>



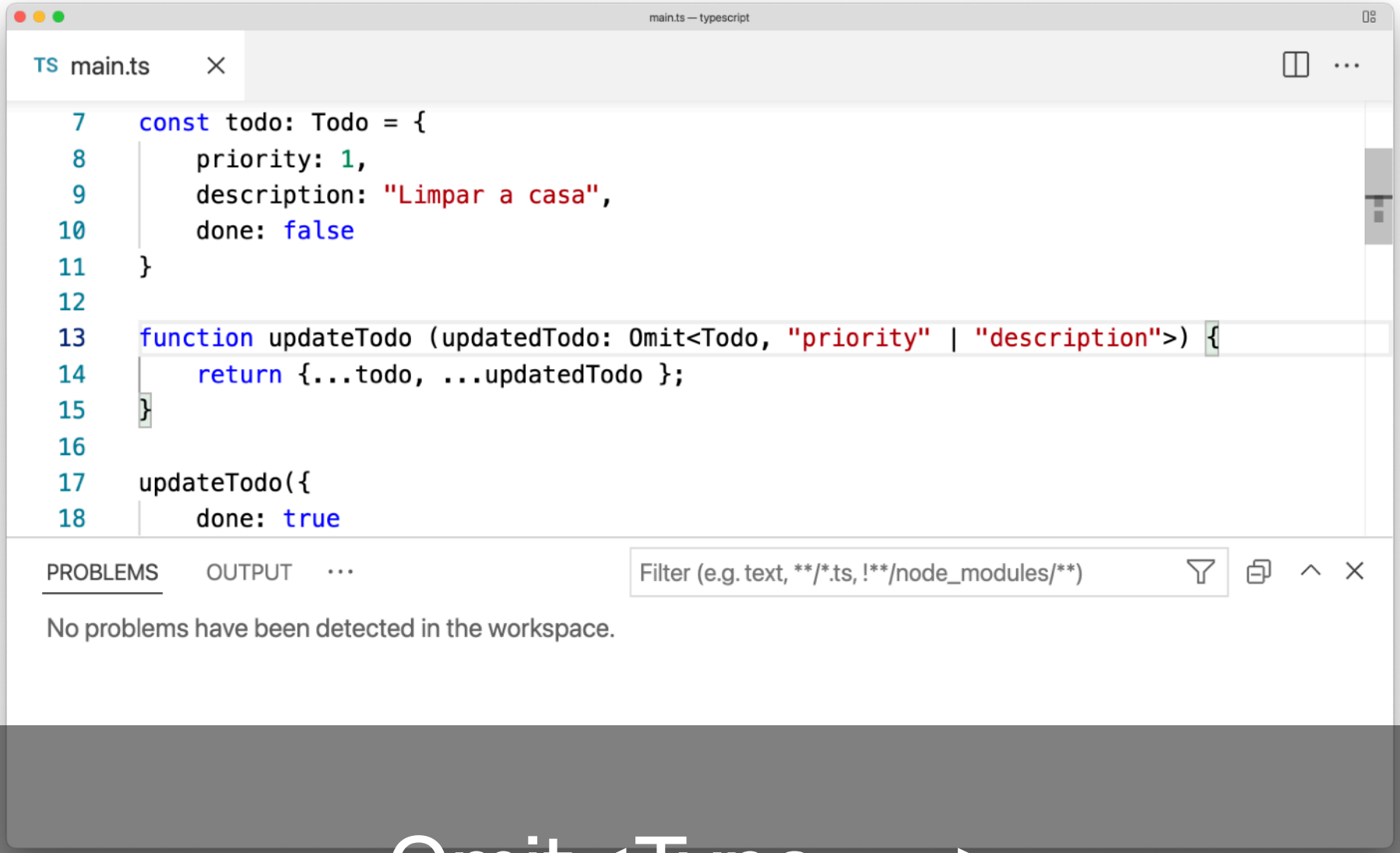
Partial<Type>



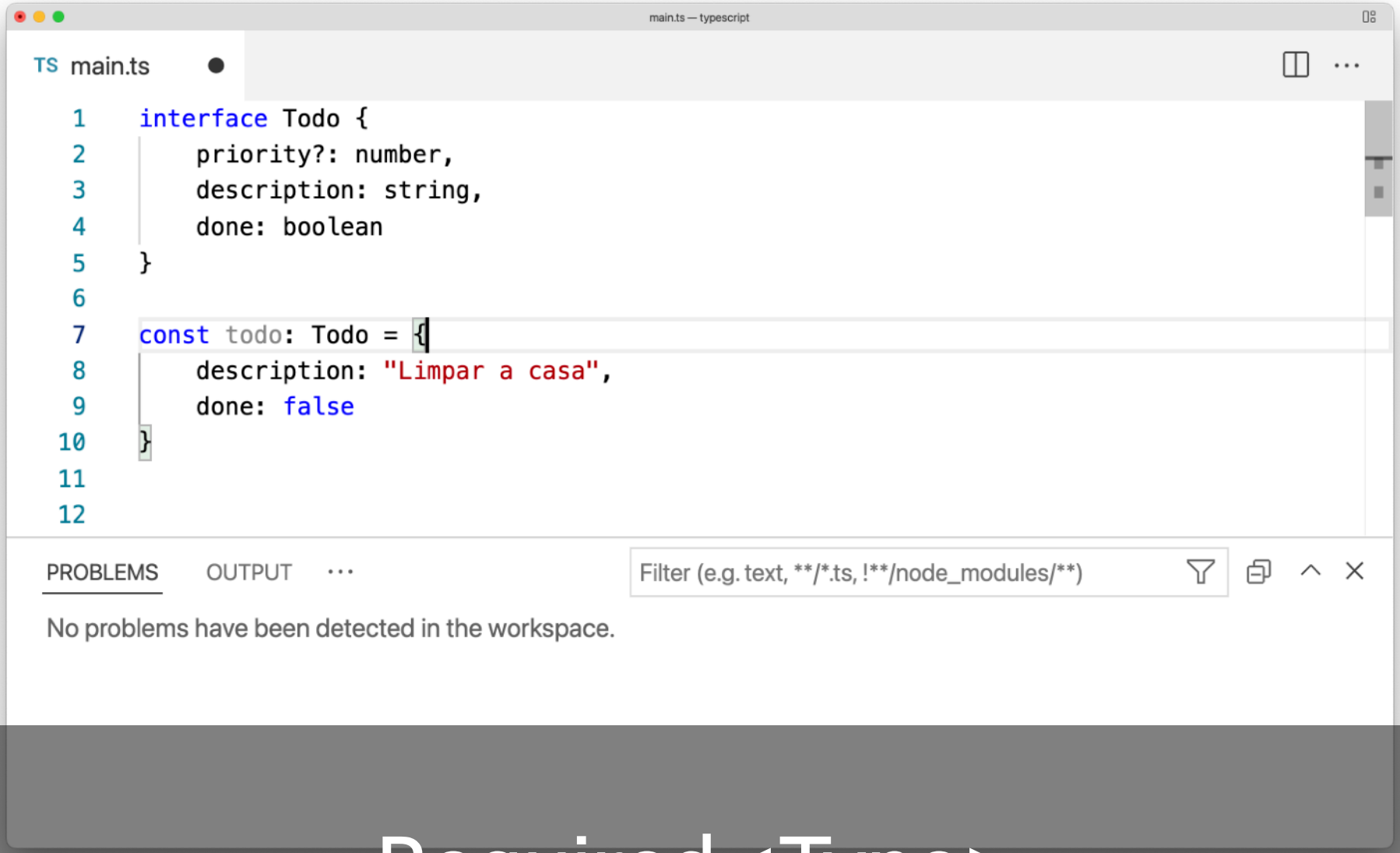
Partial<Type>


main.ts — typescript
TS main.ts
12
13 function updateTodo (updatedTodo: Pick<Todo, "done">) {
14 return {...todo, ...updatedTodo };
15 }
16
17 updateTodo({
18 done: true
19 });
20
21
22
23
PROBLEMS
OUTPUT
Filter (e.g. text, **/*.ts, !**/node_modules/**)
No problems have been detected in the workspace.

Pick<Type, ...>

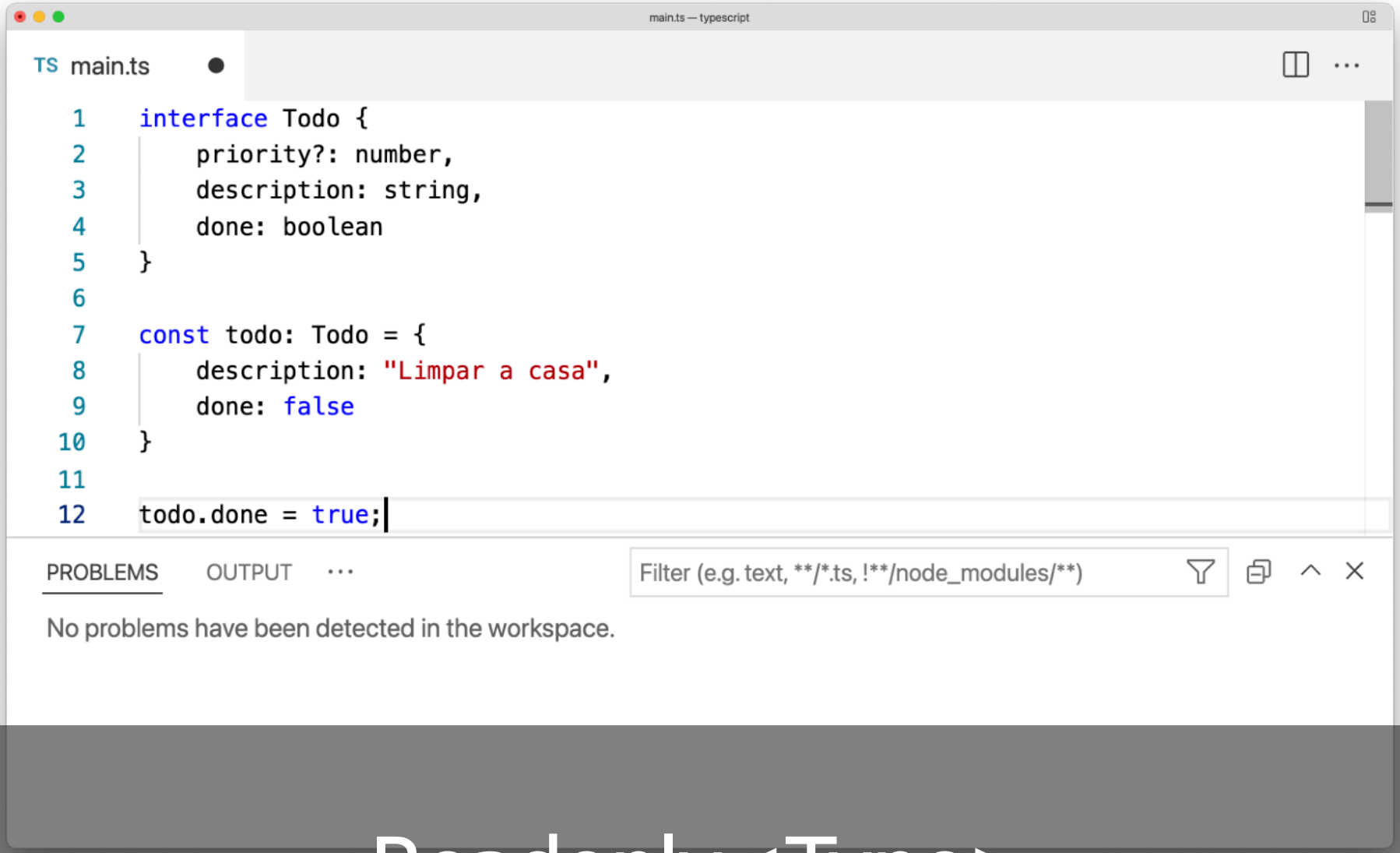

main.ts — typescript
TS main.ts
7 const todo: Todo = {
8 priority: 1,
9 description: "Limpar a casa",
10 done: false
11 }
12
13 function updateTodo (updatedTodo: Omit<Todo, "priority" | "description">) {
14 return {...todo, ...updatedTodo };
15 }
16
17 updateTodo({
18 done: true
PROBLEMS
OUTPUT
Filter (e.g. text, **/*.ts, !**/node_modules/**)
No problems have been detected in the workspace.

Omit<Type, ...>

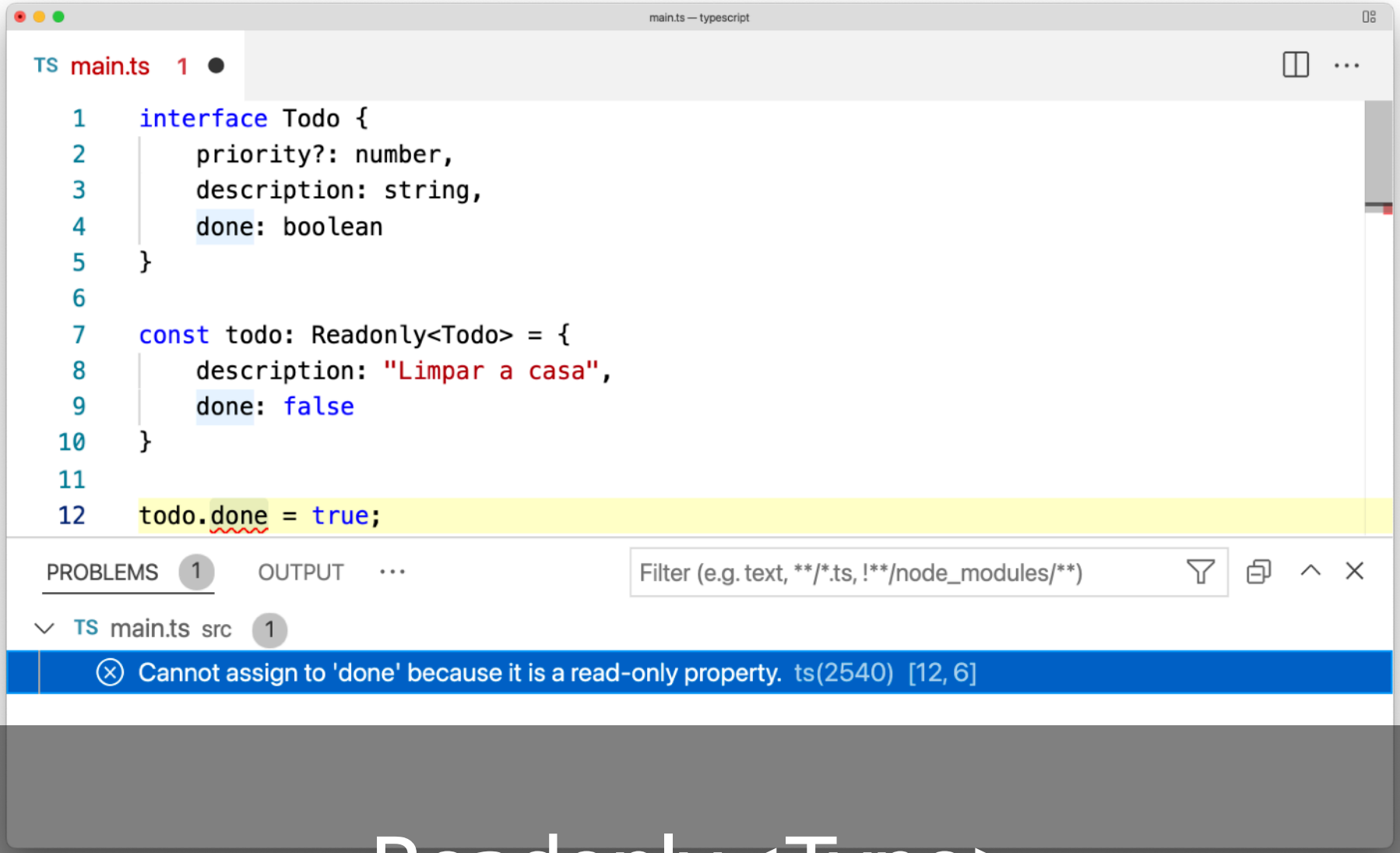


Required<Type>

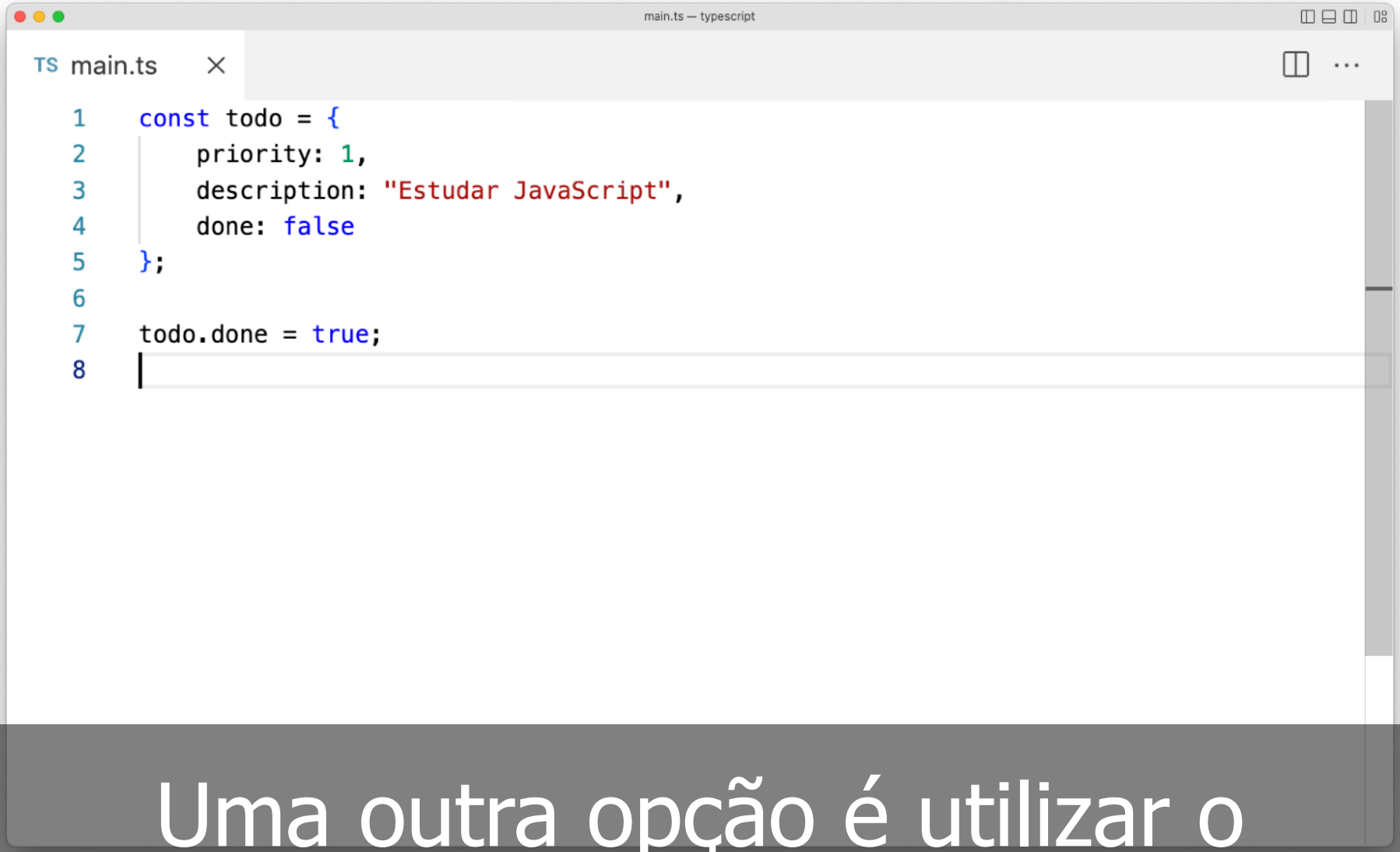
Required<Type>



Readonly<Type>

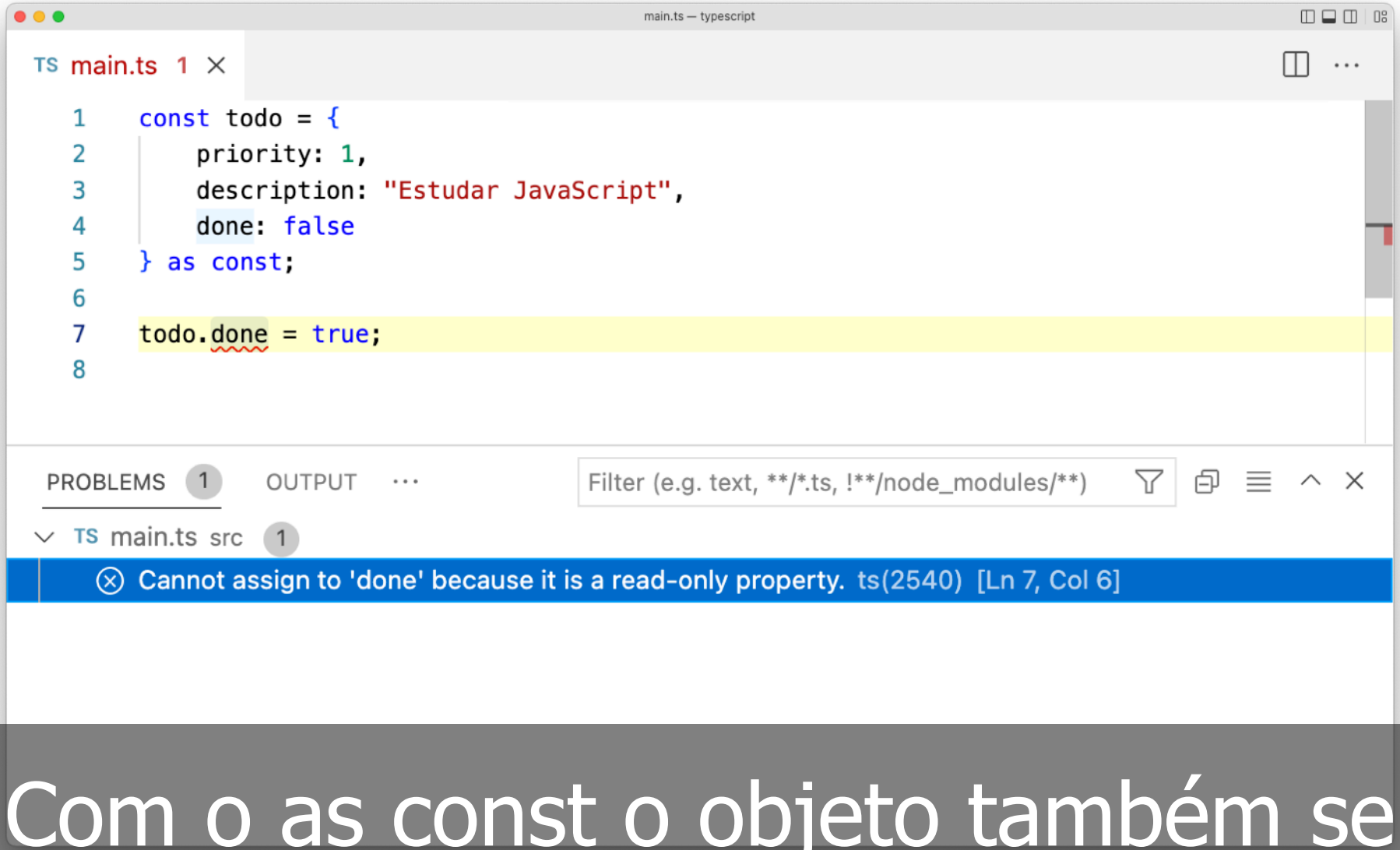


Readonly<Type>



Uma outra opção é utilizar o

as const



Com o as const o objeto também se torna imutável, somente leitura

Existem diversos outros utility types como Exclude, Extract, NonNullable, Parameters entre outros

Type Declaration

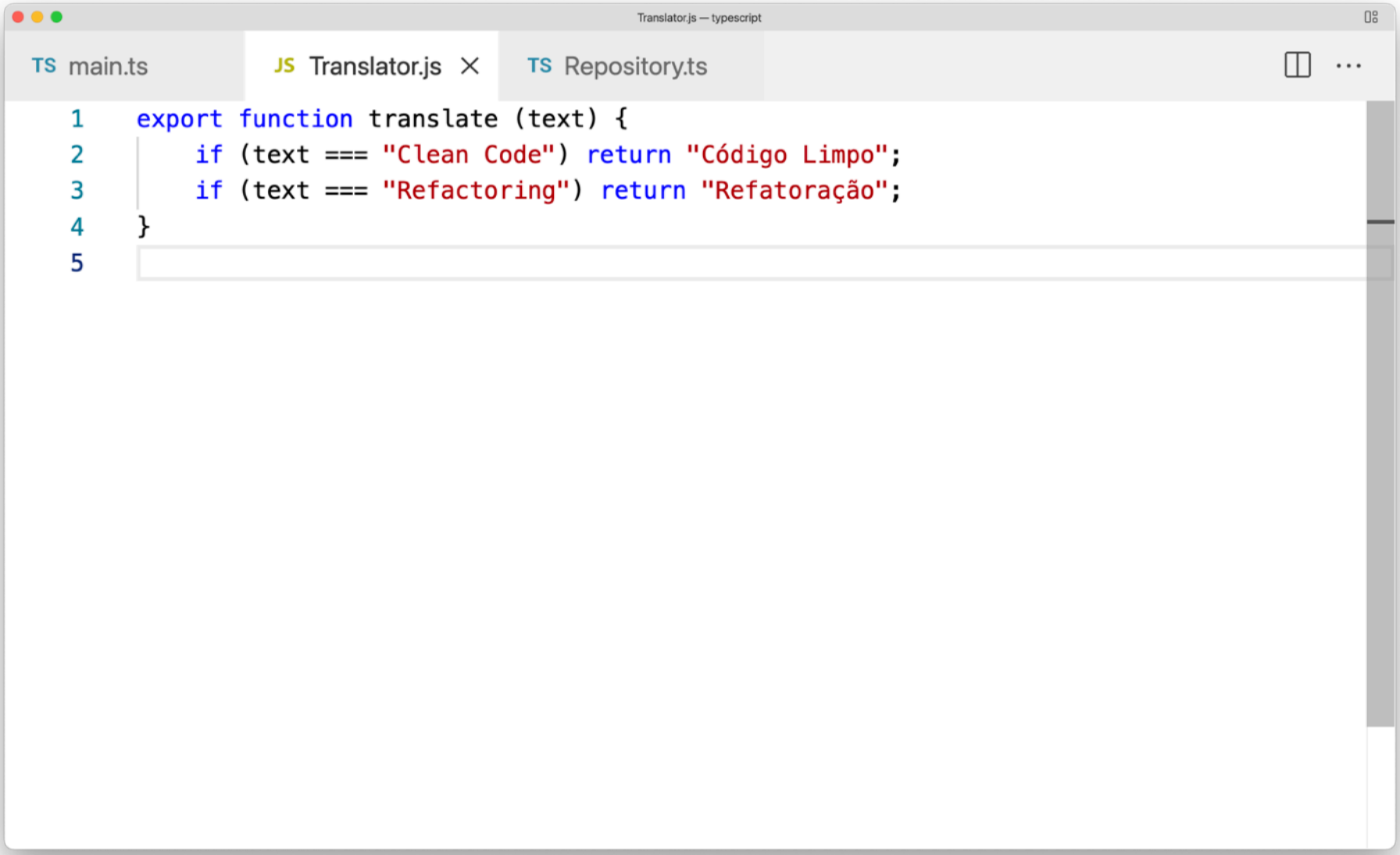
O TypeScript tem dois tipos de arquivos, .ts e .d.ts sendo que este é utilizado para a declaração de tipos, fundamental na interoperabilidade com dependências na linguagem JavaScript

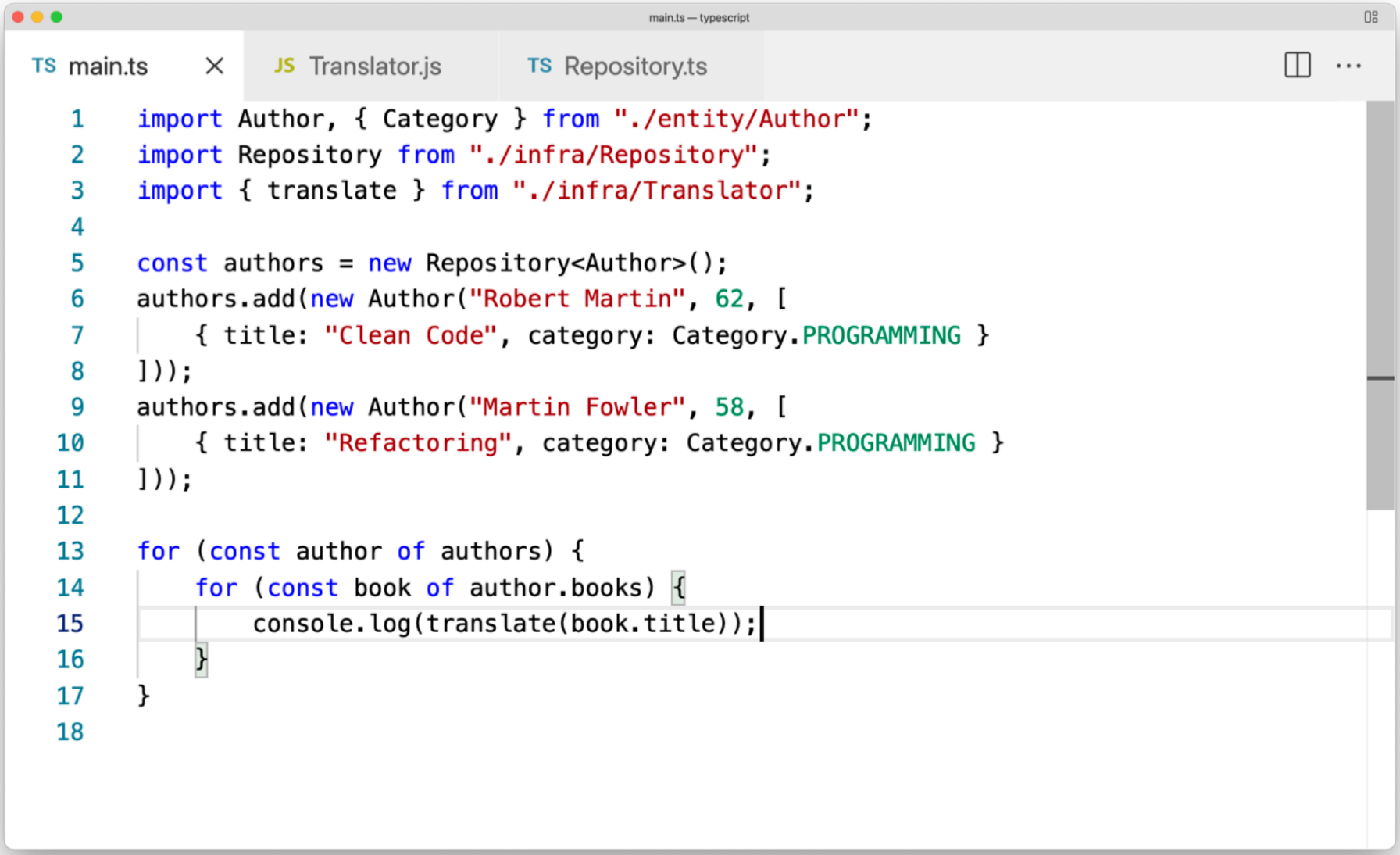
JavaScript Support

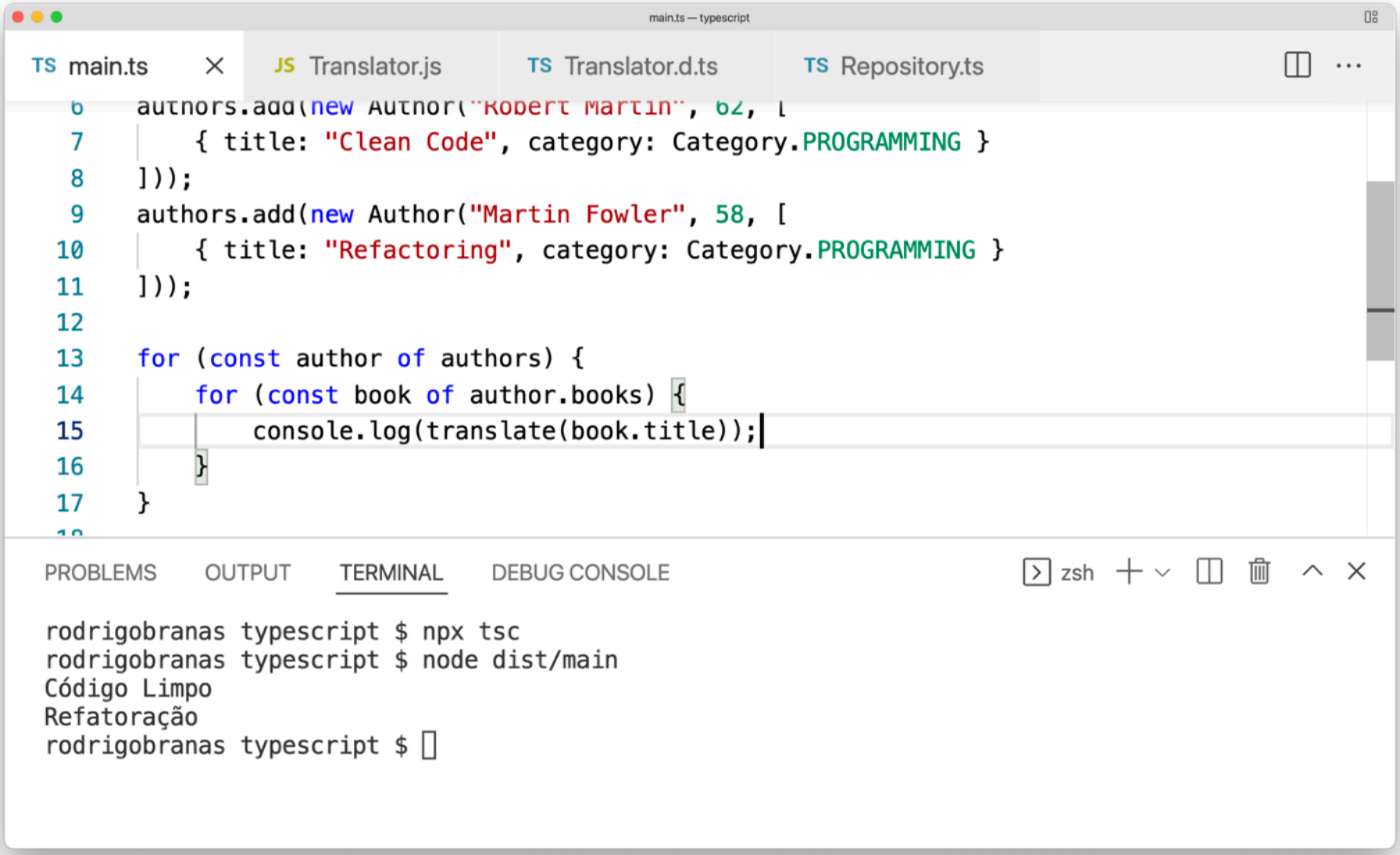
O grupo de configurações JavaScript Support é utilizado para habilitar e validar a interoperabilidade com a linguagem JavaScript, copiando os arquivos para o outFile e outDir ou checando questões relacionadas a tipagem ou mesmo das bibliotecas que estão sendo utilizadas

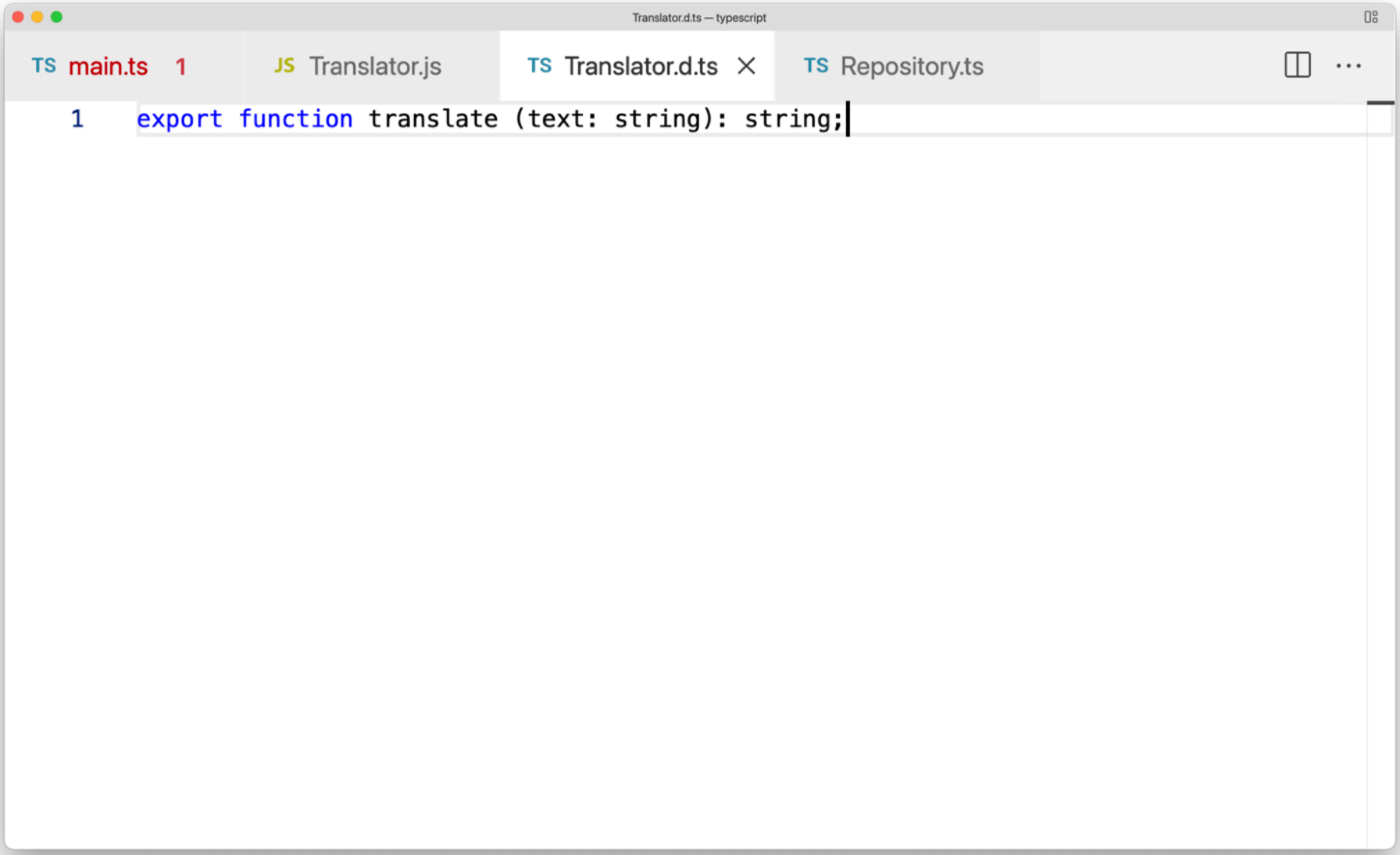
A opção allowJs habilita a utilização da linguagem JavaScript em conjunto com TypeScript, incorporando ambas na pasta de destino

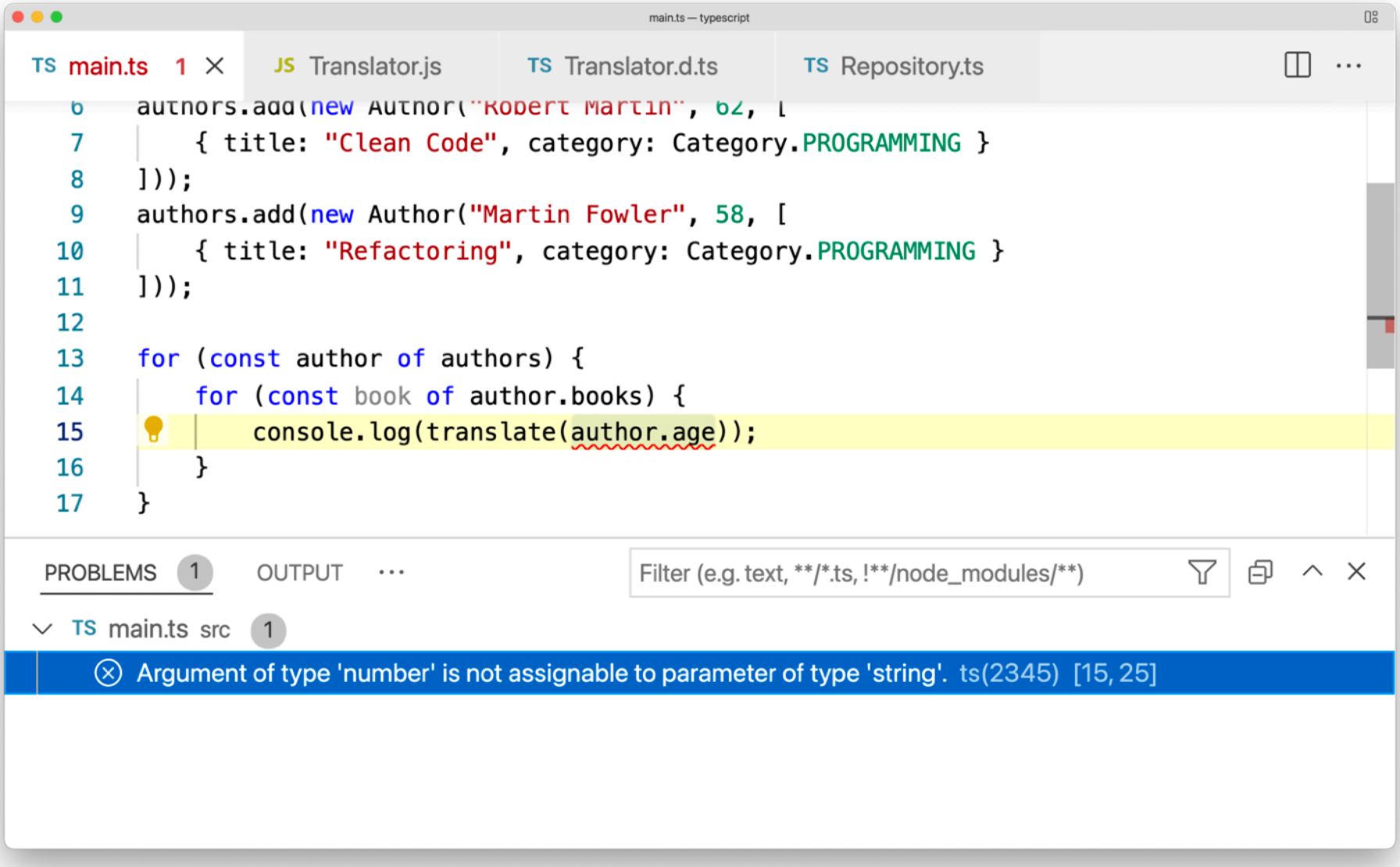
A opção checkJs realiza uma verificação em arquivos escritos na linguagem JavaScript em relação a tipos, parâmetros na invocação de funções e muito mais


Translator.js — typescript
TS main.ts
JS Translator.js
TS Repository.ts
export function translate (text) {
if (text === "Clean Code") return "Código Limpo";
if (text === "Refactoring") return "Refatoração";
}


main.ts — typescript
TS main.ts
JS Translator.js
TS Repository.ts
import Author, { Category } from "./entity/Author";
import Repository from "./infra/Repository";
import { translate } from "./infra/Translator";
const authors = new Repository<Author>();
authors.add(new Author("Robert Martin", 62, [
{ title: "Clean Code", category: Category.PROGRAMMING }
]));
authors.add(new Author("Martin Fowler", 58, [
{ title: "Refactoring", category: Category.PROGRAMMING }
]));
for (const author of authors) {
for (const book of author.books) {
console.log(translate(book.title));
}
}


main.ts — typescript
TS main.ts
JS Translator.js
TS Translator.d.ts
TS Repository.ts
authors.add(new Author("Robert Martin", 62, [
{ title: "Clean Code", category: Category.PROGRAMMING }
]));
authors.add(new Author("Martin Fowler", 58, [
{ title: "Refactoring", category: Category.PROGRAMMING }
]));
for (const author of authors) {
for (const book of author.books) {
console.log(translate(book.title));
}
}
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
zsh
rodrigobranas typescript $ npx tsc
rodrigobranas typescript $ node dist/main
Código Limpo
Refatoração
rodrigobranas typescript $


Translator.d.ts — typescript
TS main.ts 1
JS Translator.js
TS Translator.d.ts
TS Repository.ts
1 export function translate (text: string): string;


main.ts — typescript
TS main.ts 1
JS Translator.js
TS Translator.d.ts
TS Repository.ts
authors.add(new Author("Robert Martin", 62, [
{ title: "Clean Code", category: Category.PROGRAMMING }
]));
authors.add(new Author("Martin Fowler", 58, [
{ title: "Refactoring", category: Category.PROGRAMMING }
]));
for (const author of authors) {
for (const book of author.books) {
console.log(translate(author.age));
}
}
PROBLEMS 1
OUTPUT
Filter (e.g. text, **/*.ts, !**/node_modules/**)
TS main.ts src 1
Argument of type 'number' is not assignable to parameter of type 'string'. ts(2345) [15, 25]

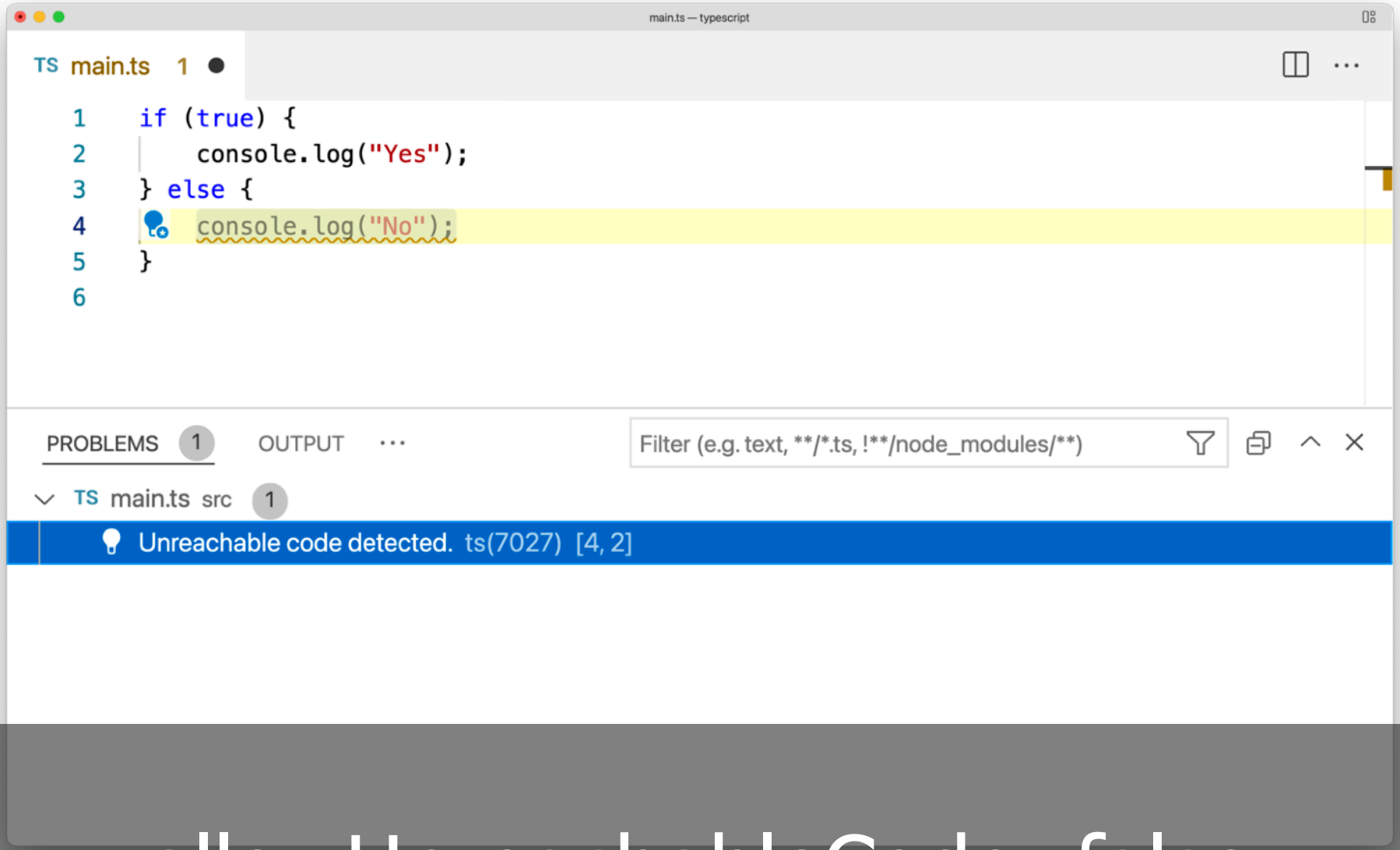
A opção declaration indica que o transpilador deve criar um arquivo .d.ts, que é o arquivo responsável pela definição dos tipos

Type Checking

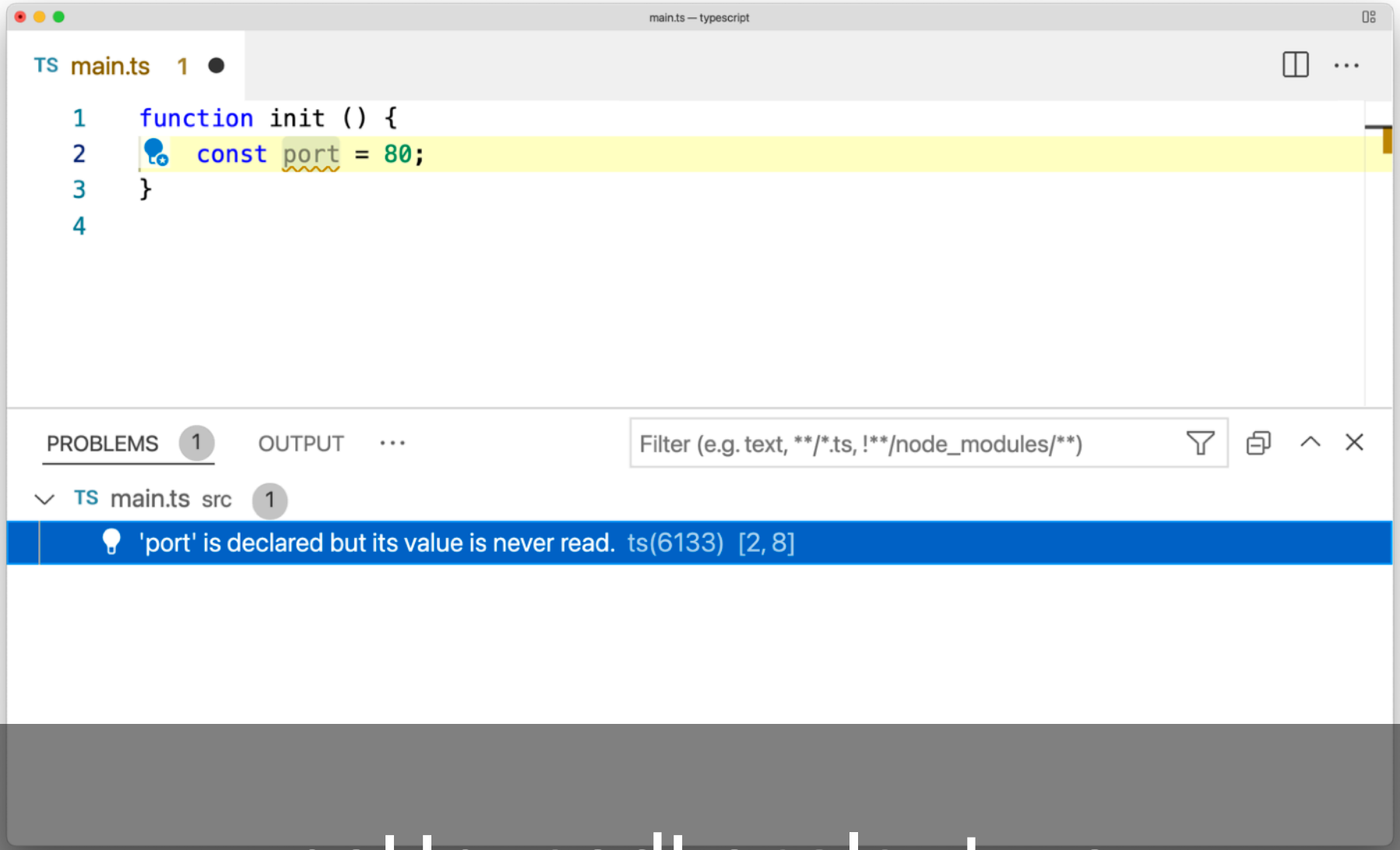
É possível configurar exatamente o comportamento das verificações refletindo as políticas estabelecidas no projeto, assim como seria feito com qualquer outro tipo de linter

A opção strict habilita diversas verificações de tipagem como:

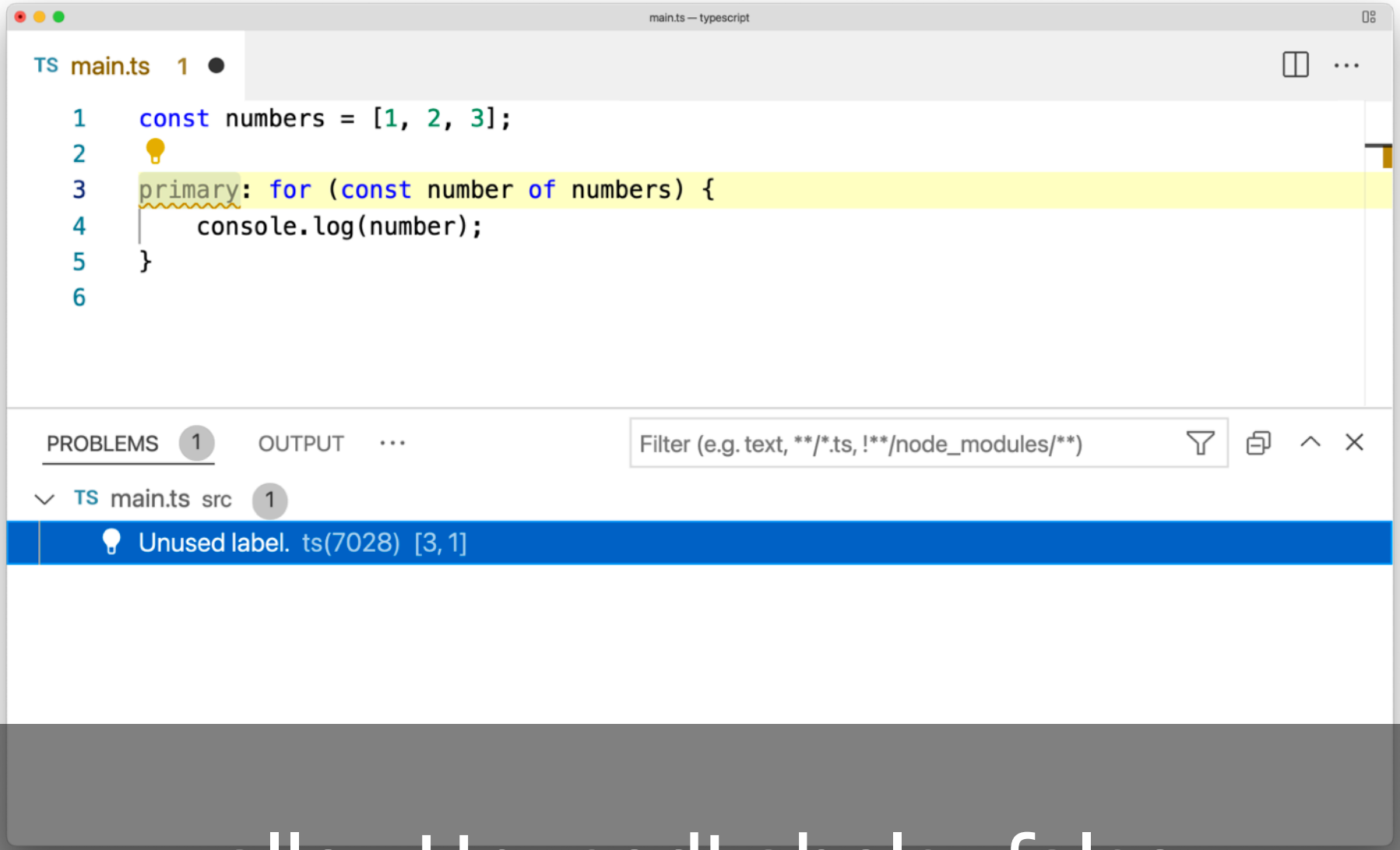
strictNullChecks
strictFunctionTypes
strictBindCallApply
strictPropertyInitialization
noImplicitAny
noImplicitThis
useUnknowInCatchVariables
alwaysStrict



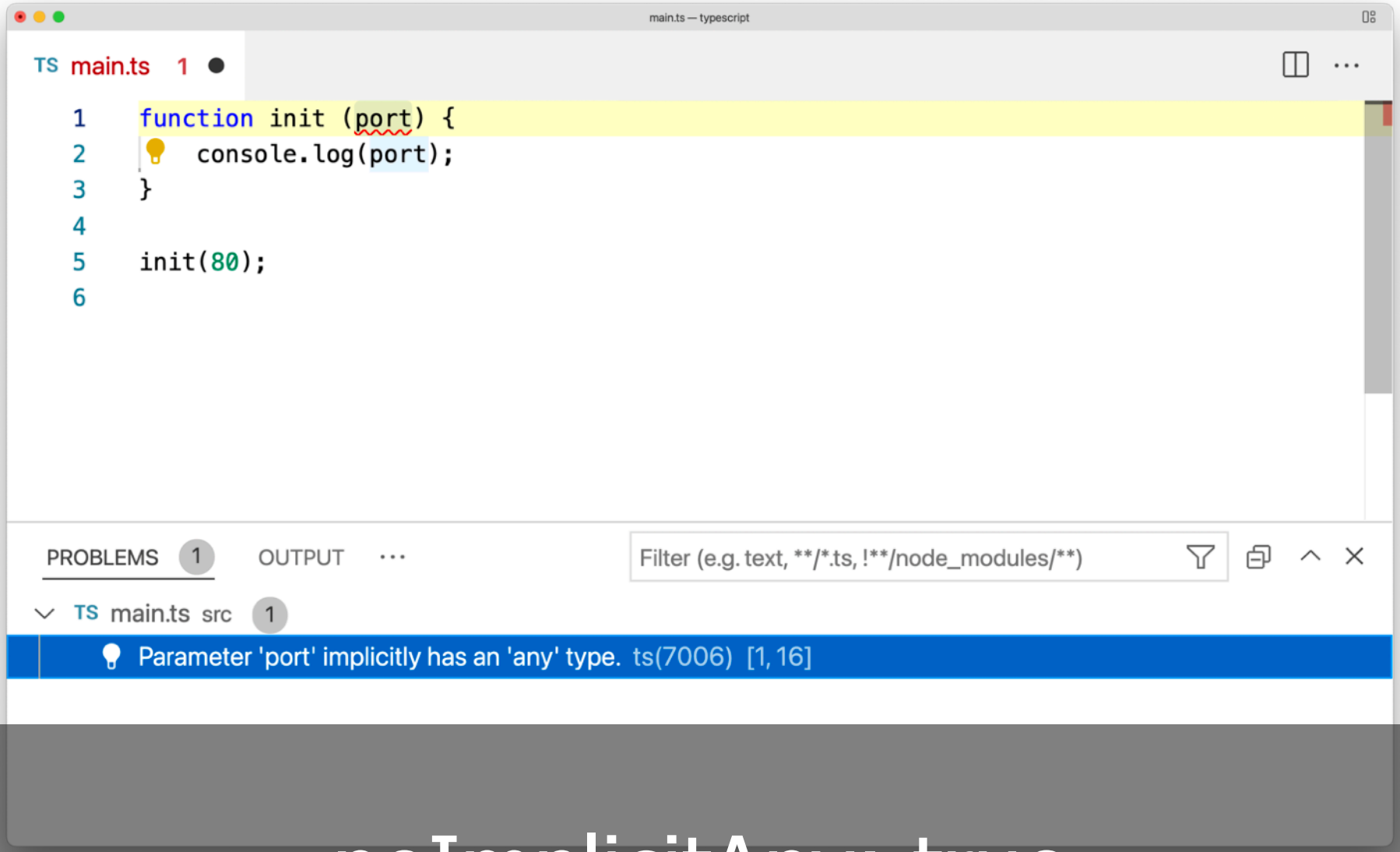
allowUnreachableCode: false



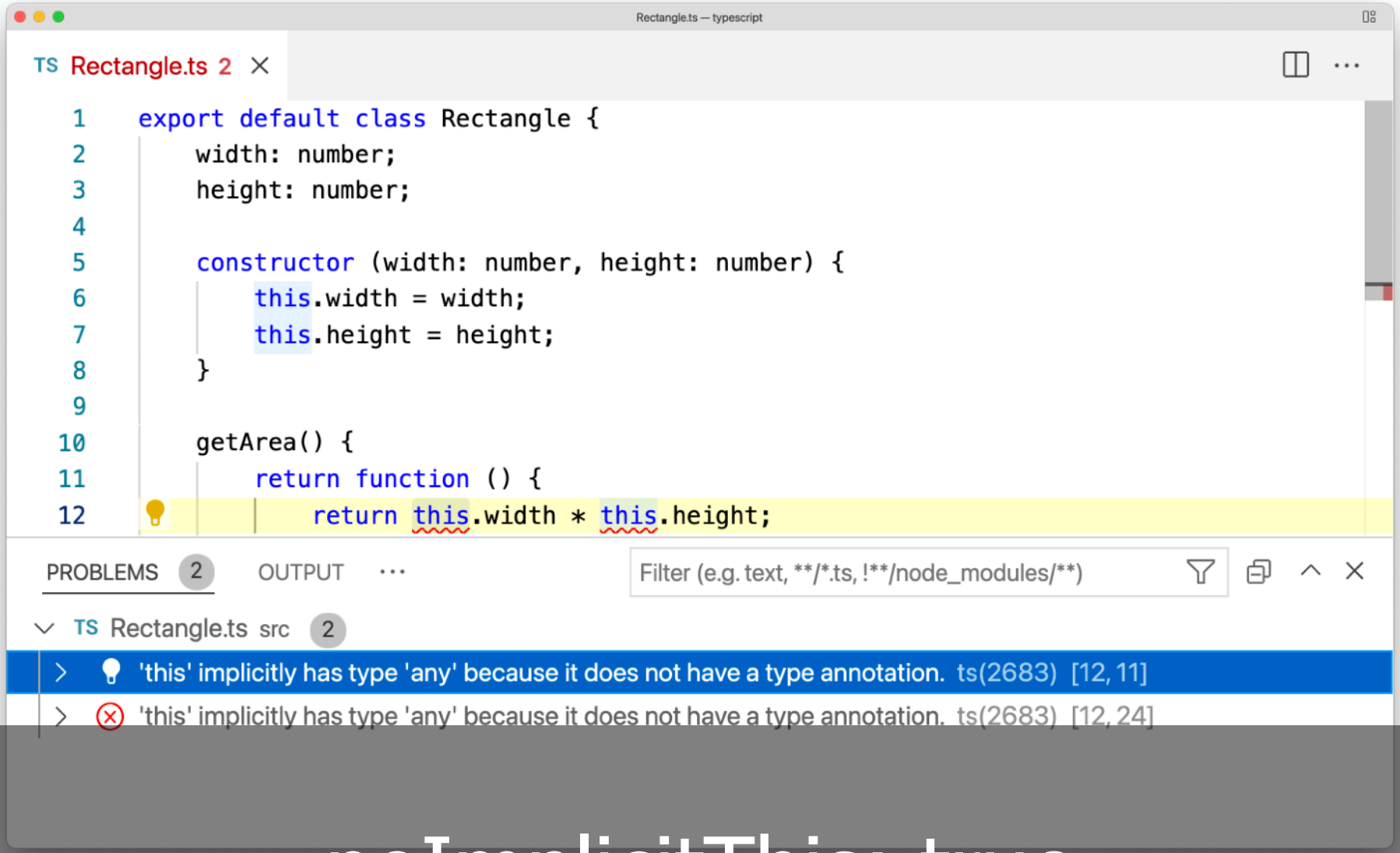
noUnusedLocals: true



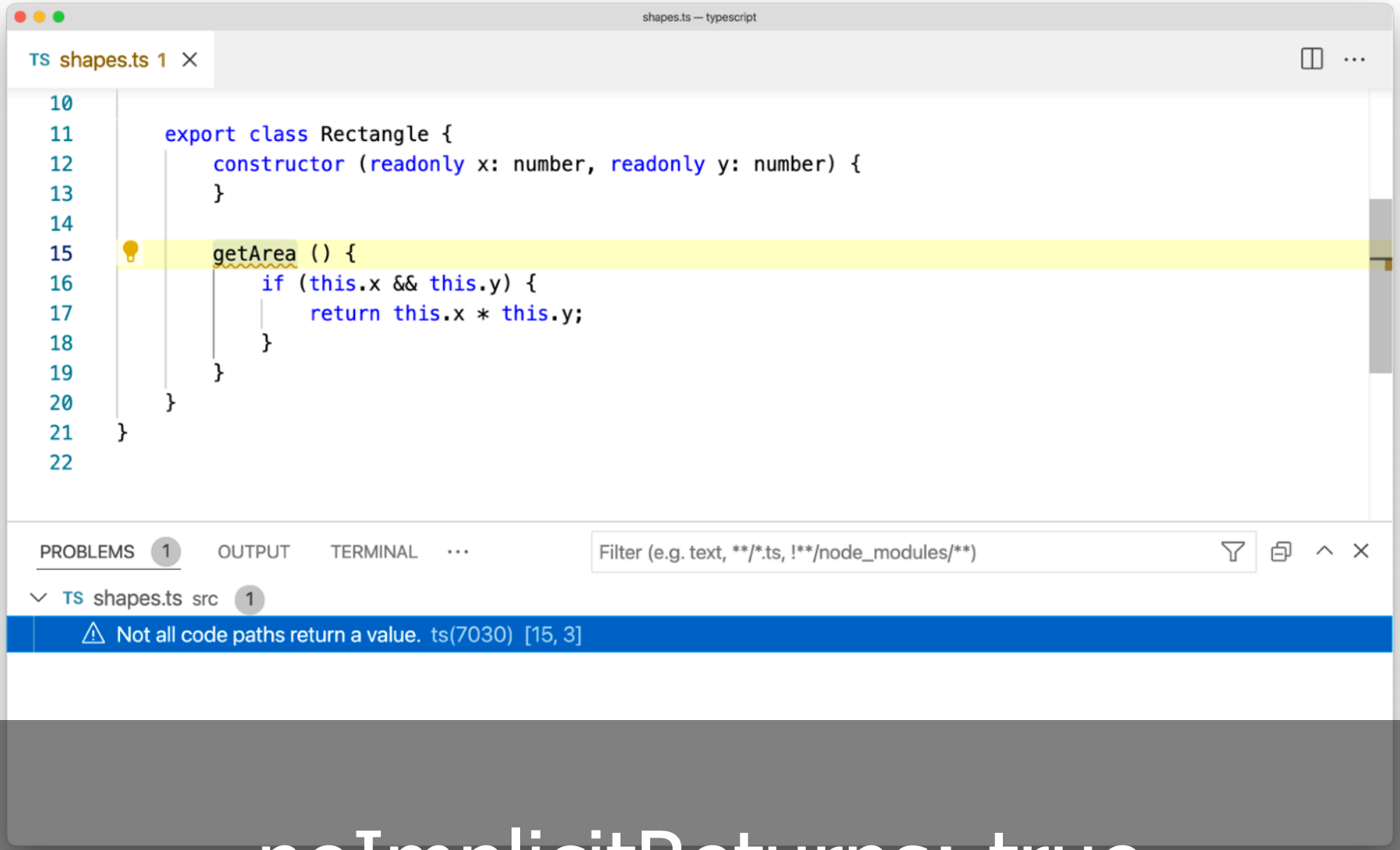
allowUnusedLabels: false



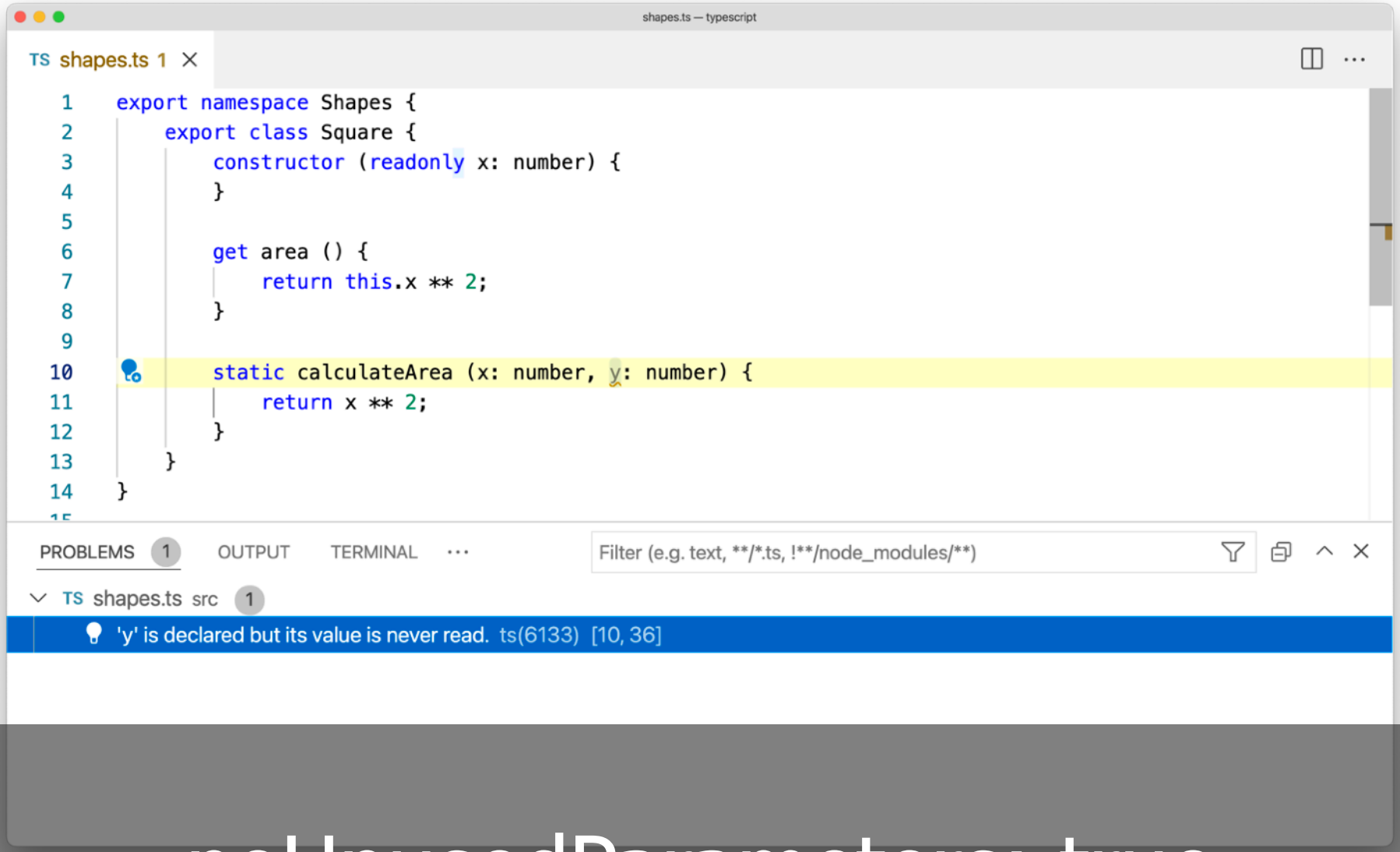
noImplicitAny: true



noImplicitThis: true

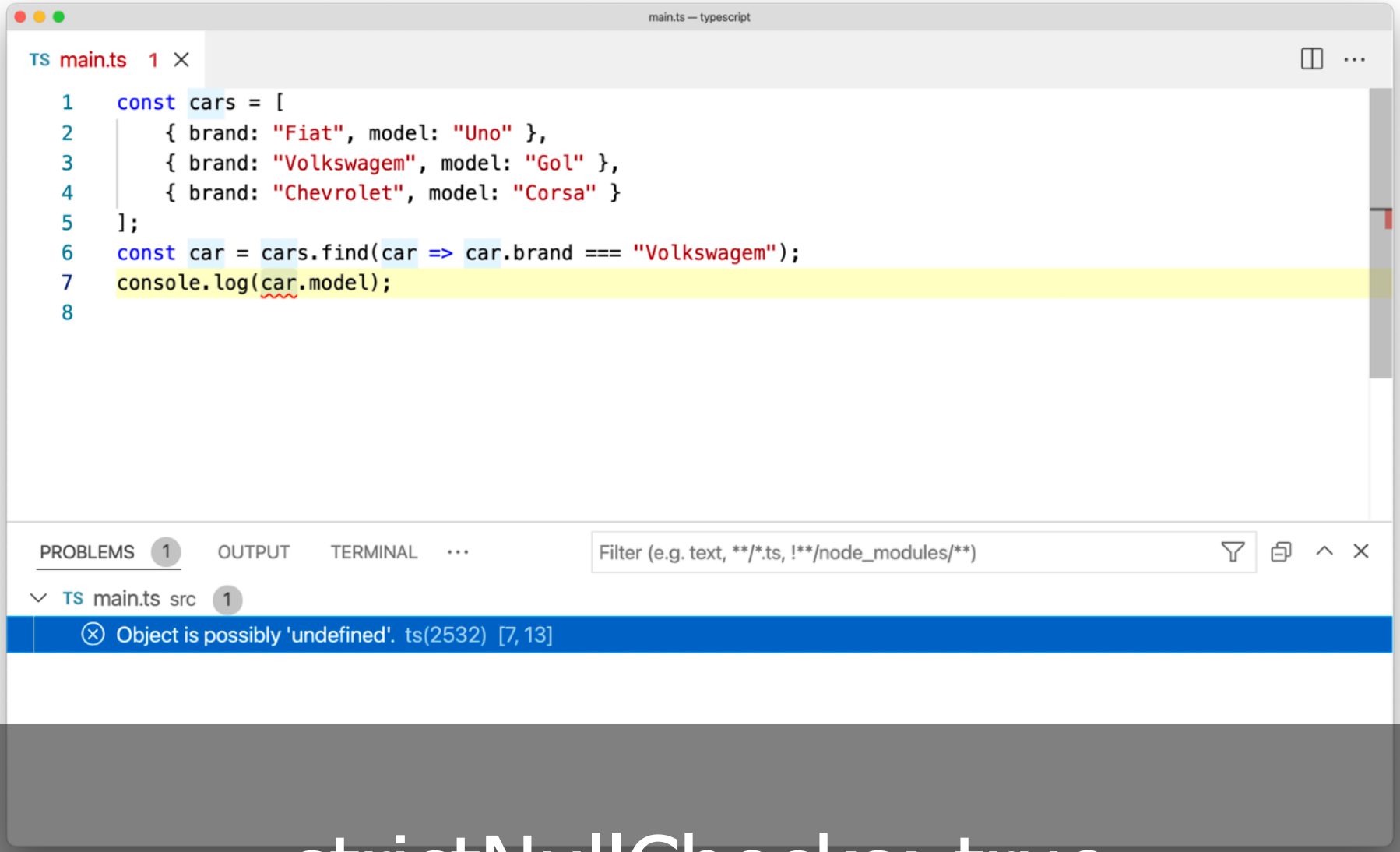


noImplicitReturns: true

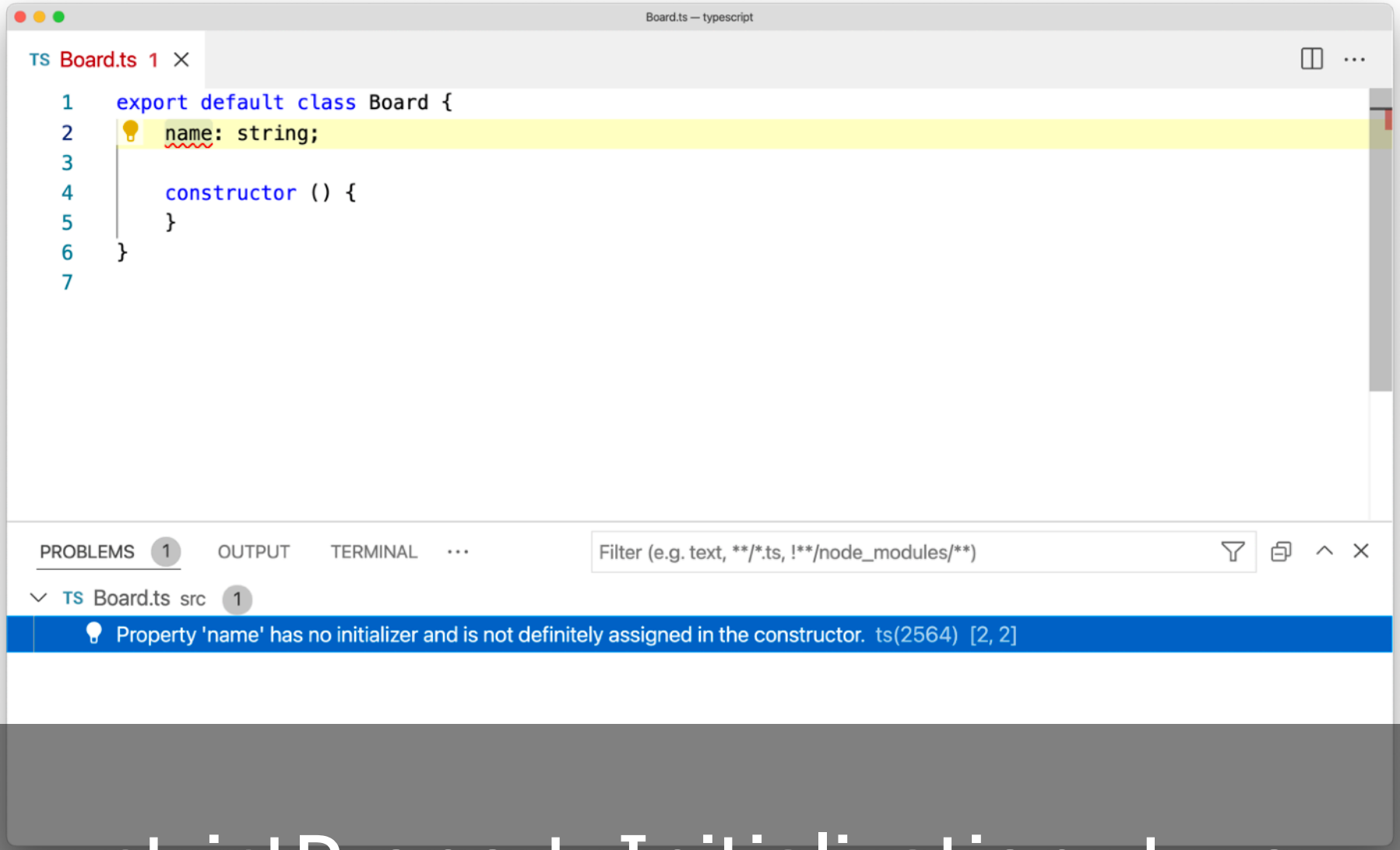


noUnusedParameters: true

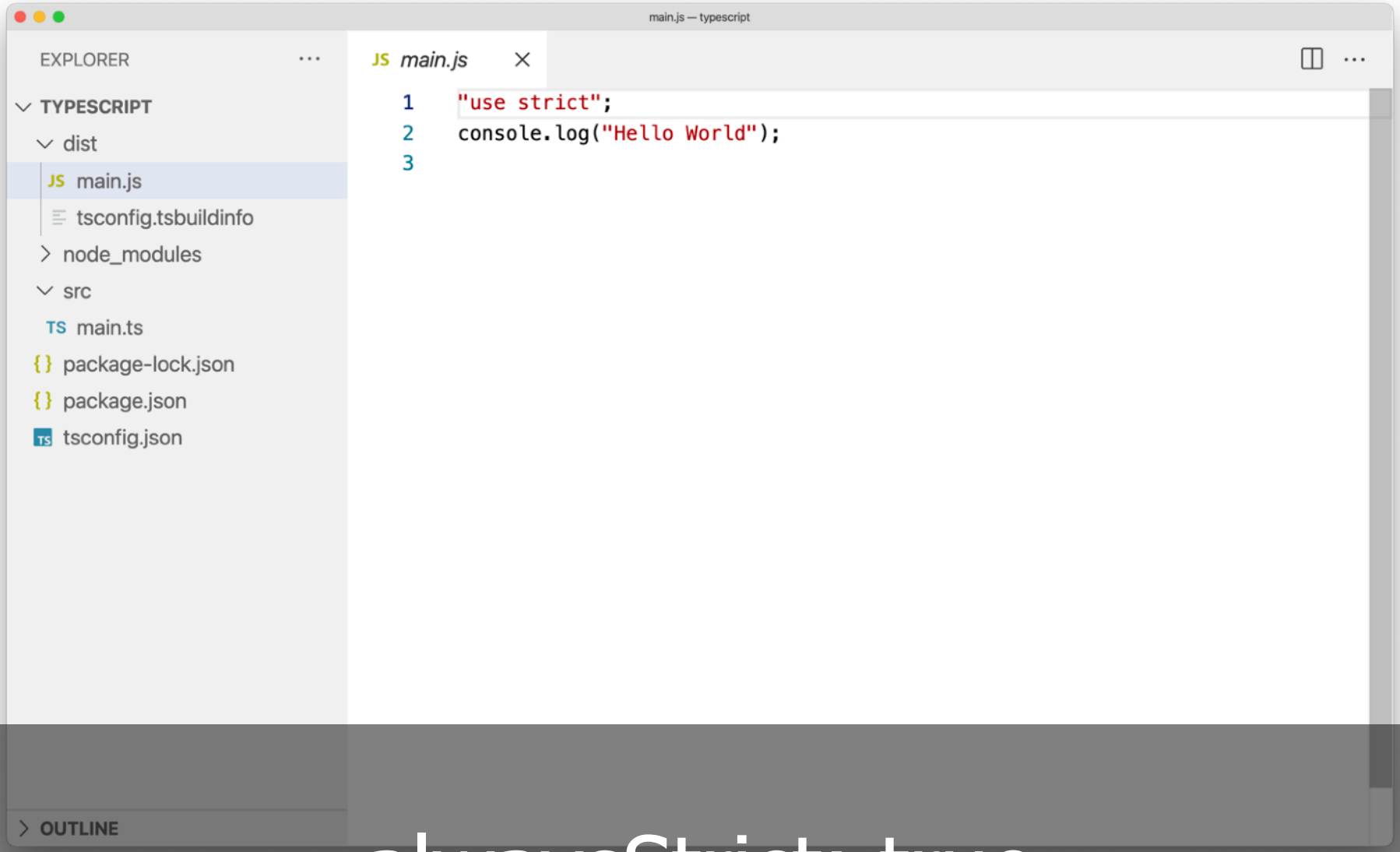
noImplicitOverride: true



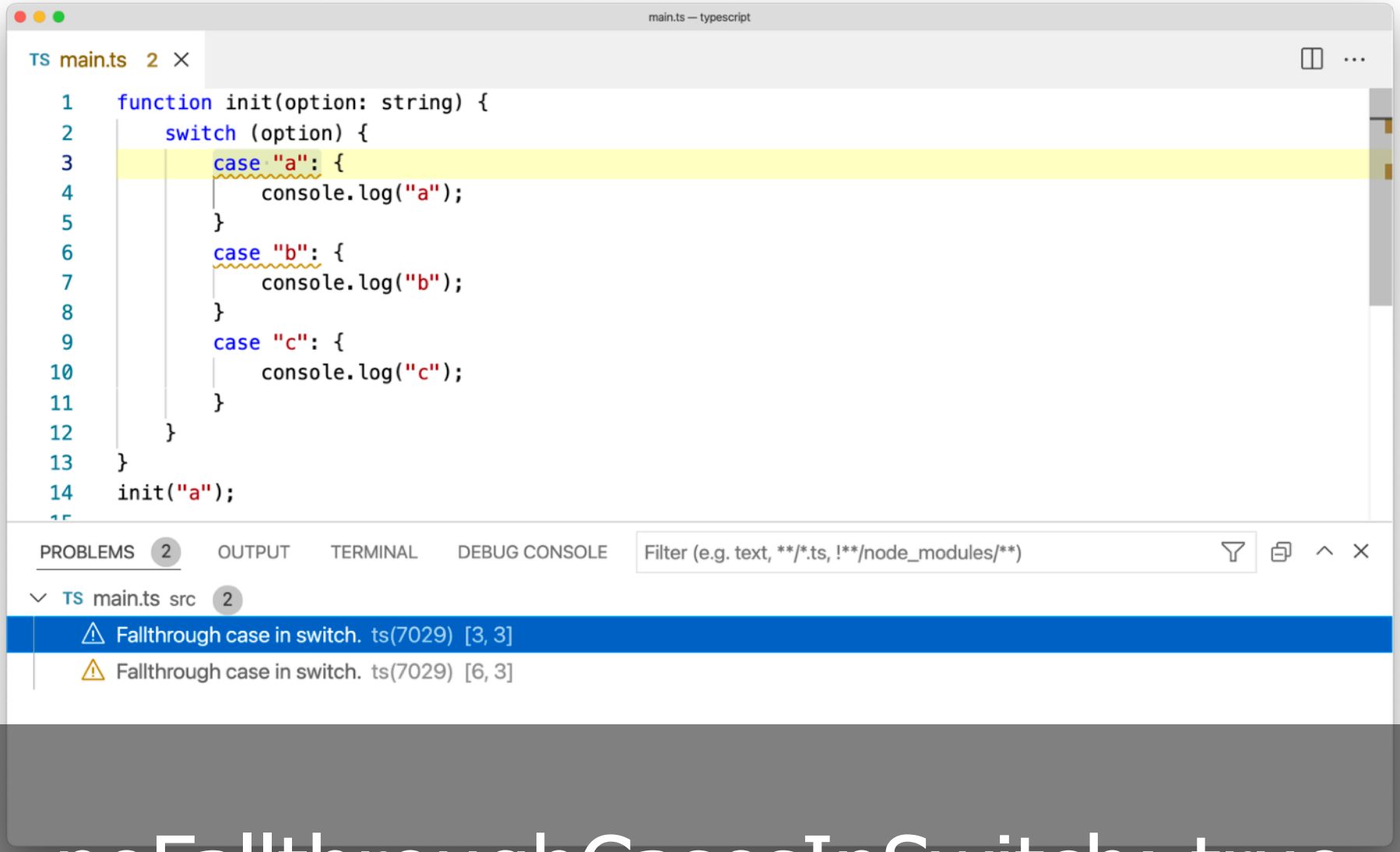
strictNullChecks: true



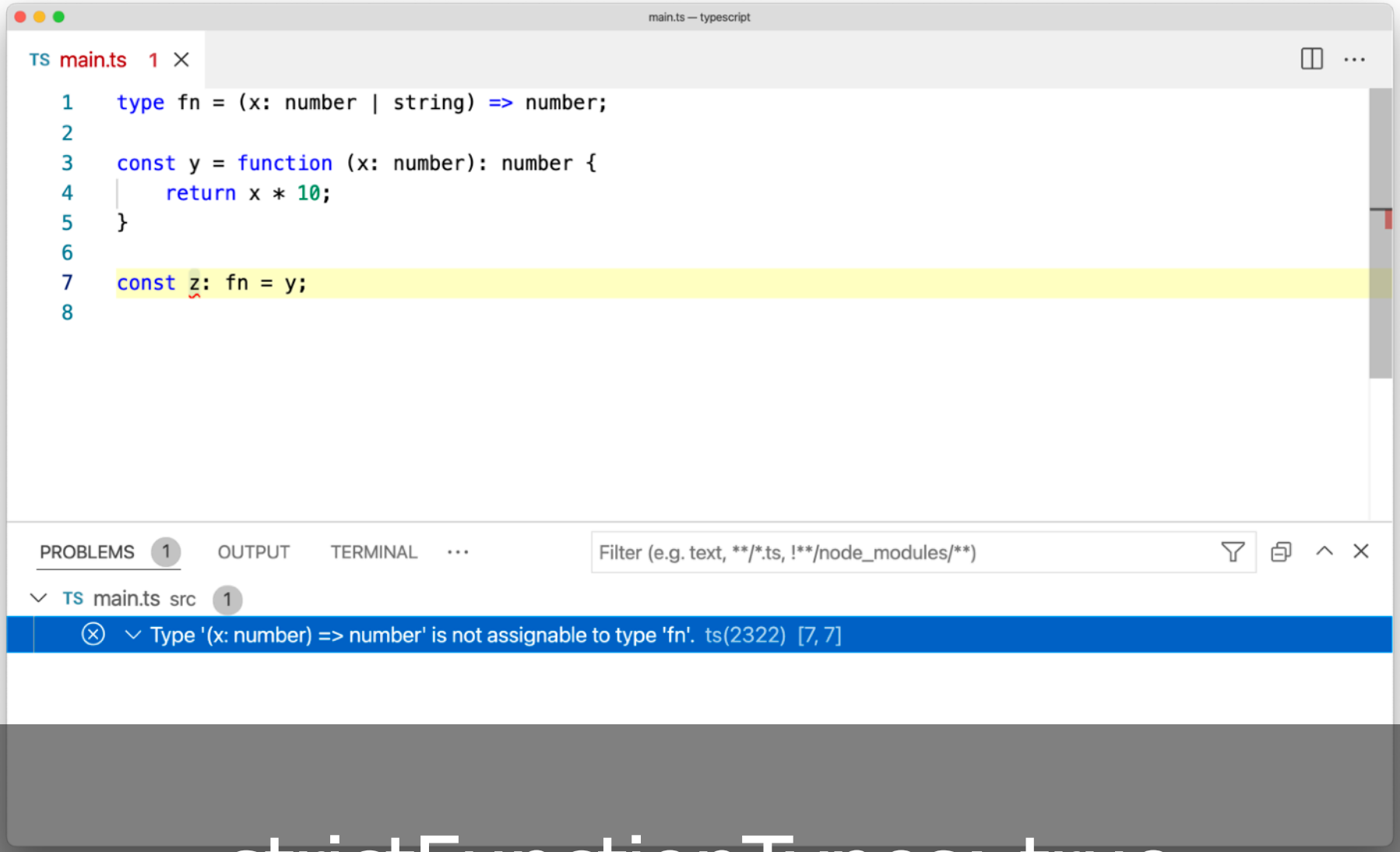
strictPropertyInitialization: true



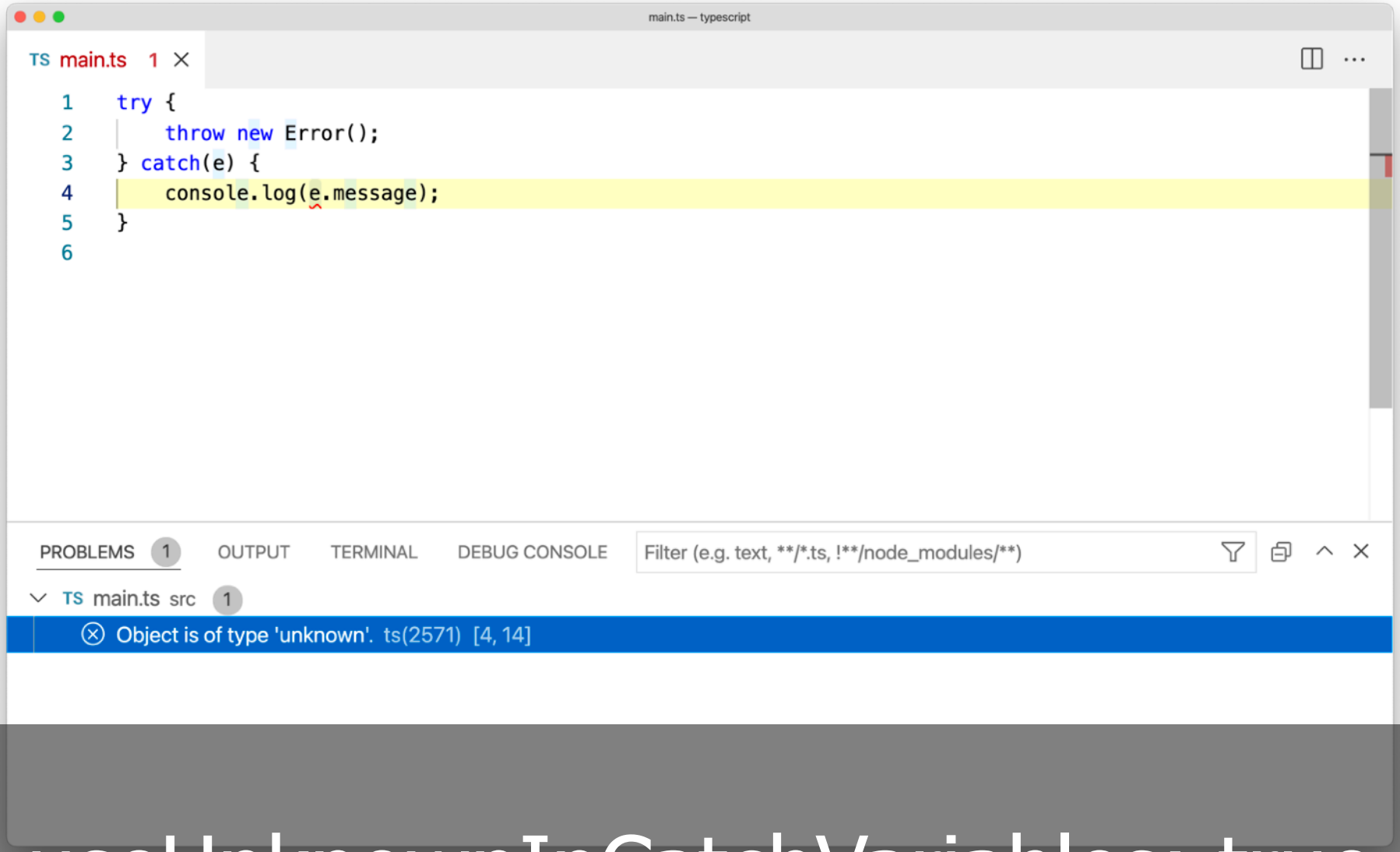
alwaysStrict: true



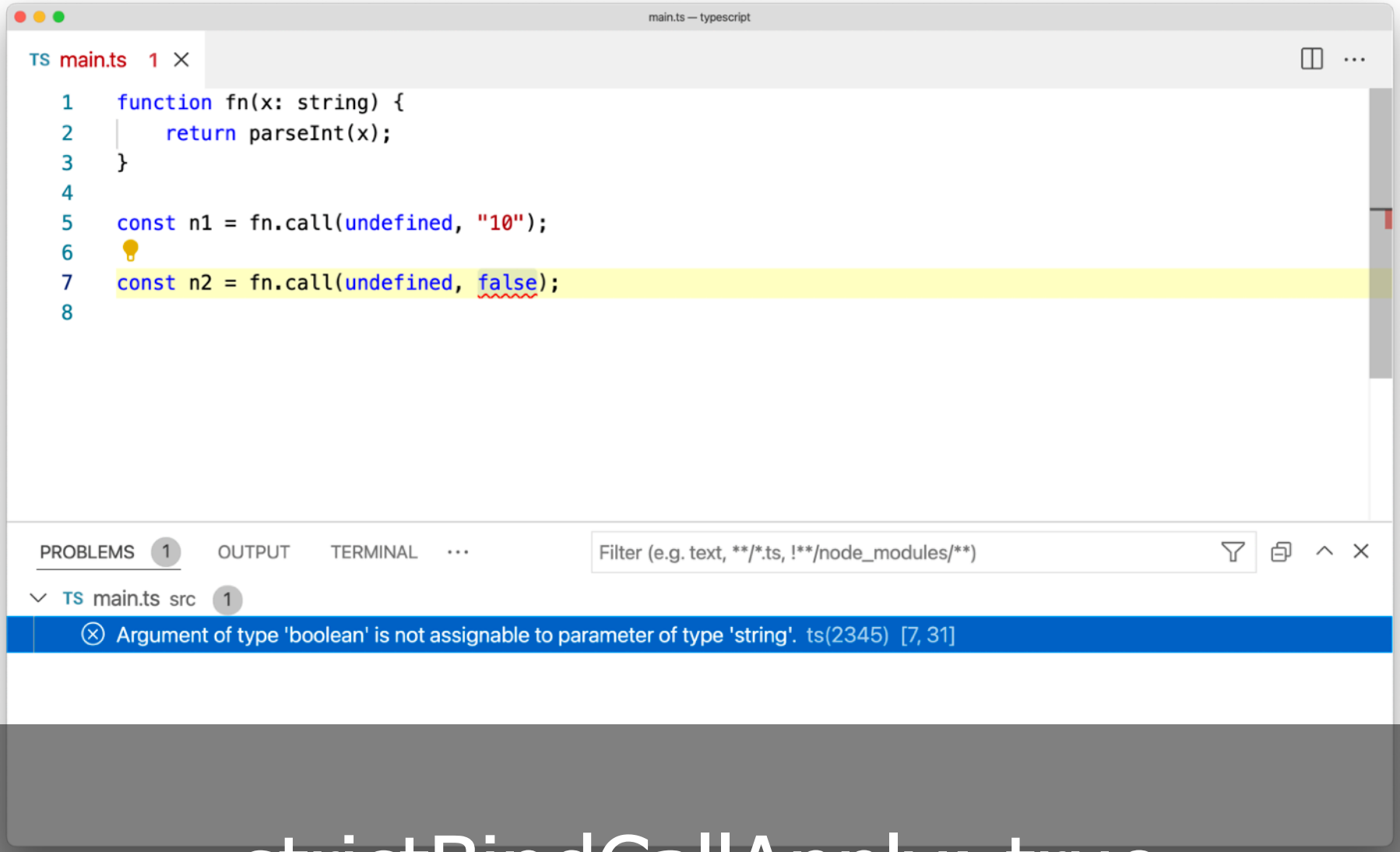
noFallthroughCasesInSwitch: true



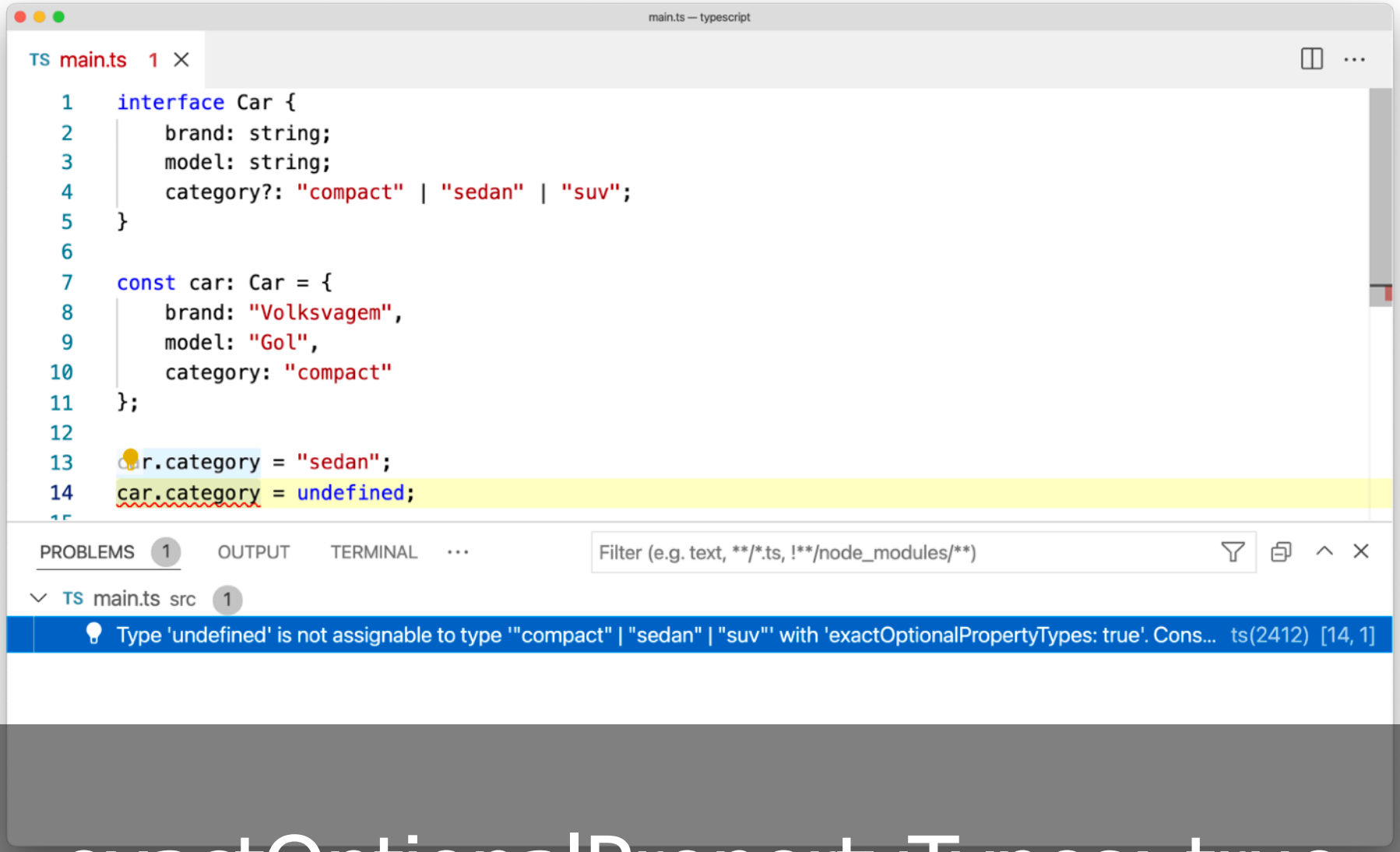
strictFunctionTypes: true



useUnknownInCatchVariables: true

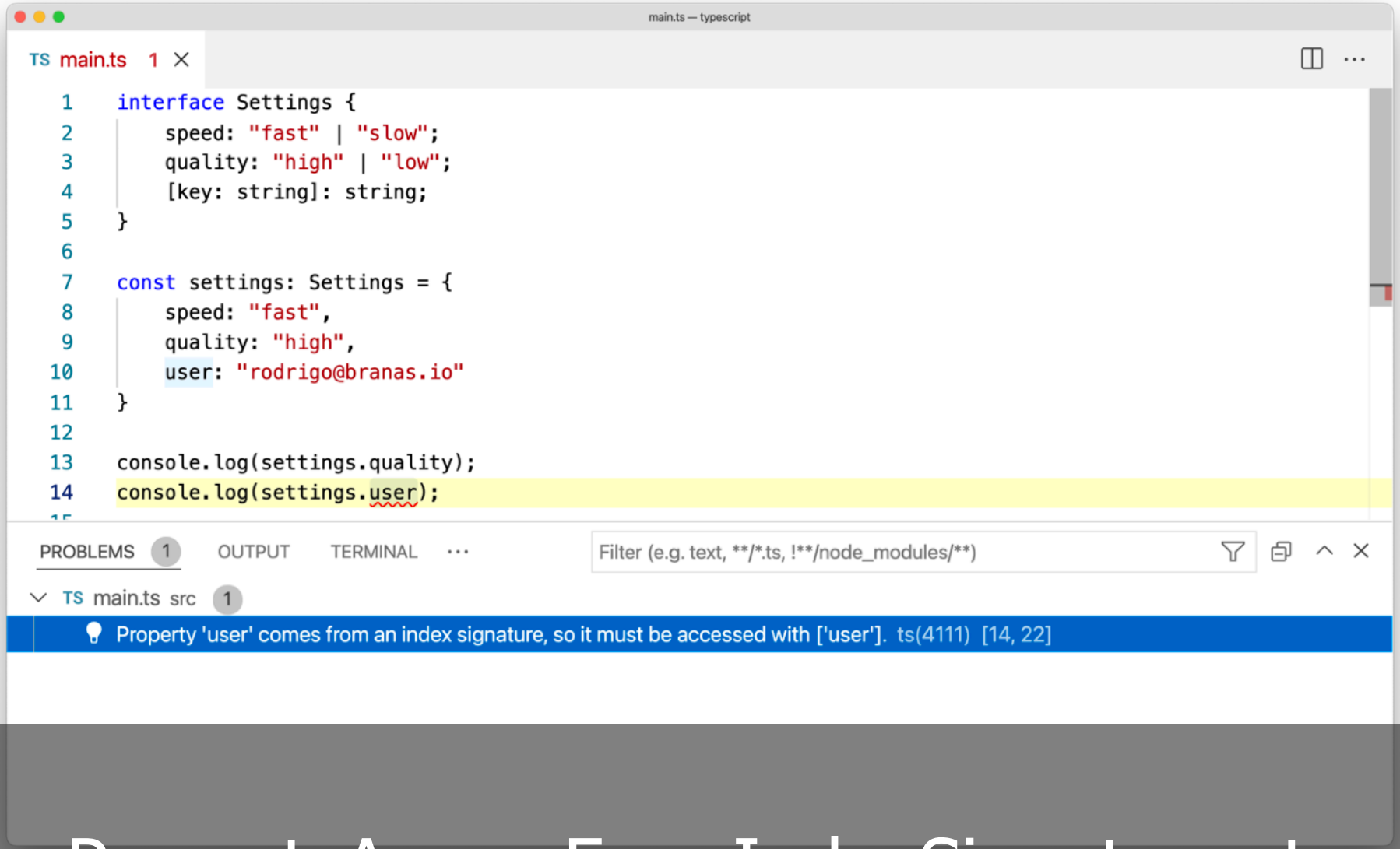


strictBindCallApply: true



exactOptionalPropertyTypes: true

noUncheckedIndexedAccess: true



noPropertyAccessFromIndexSignature: true