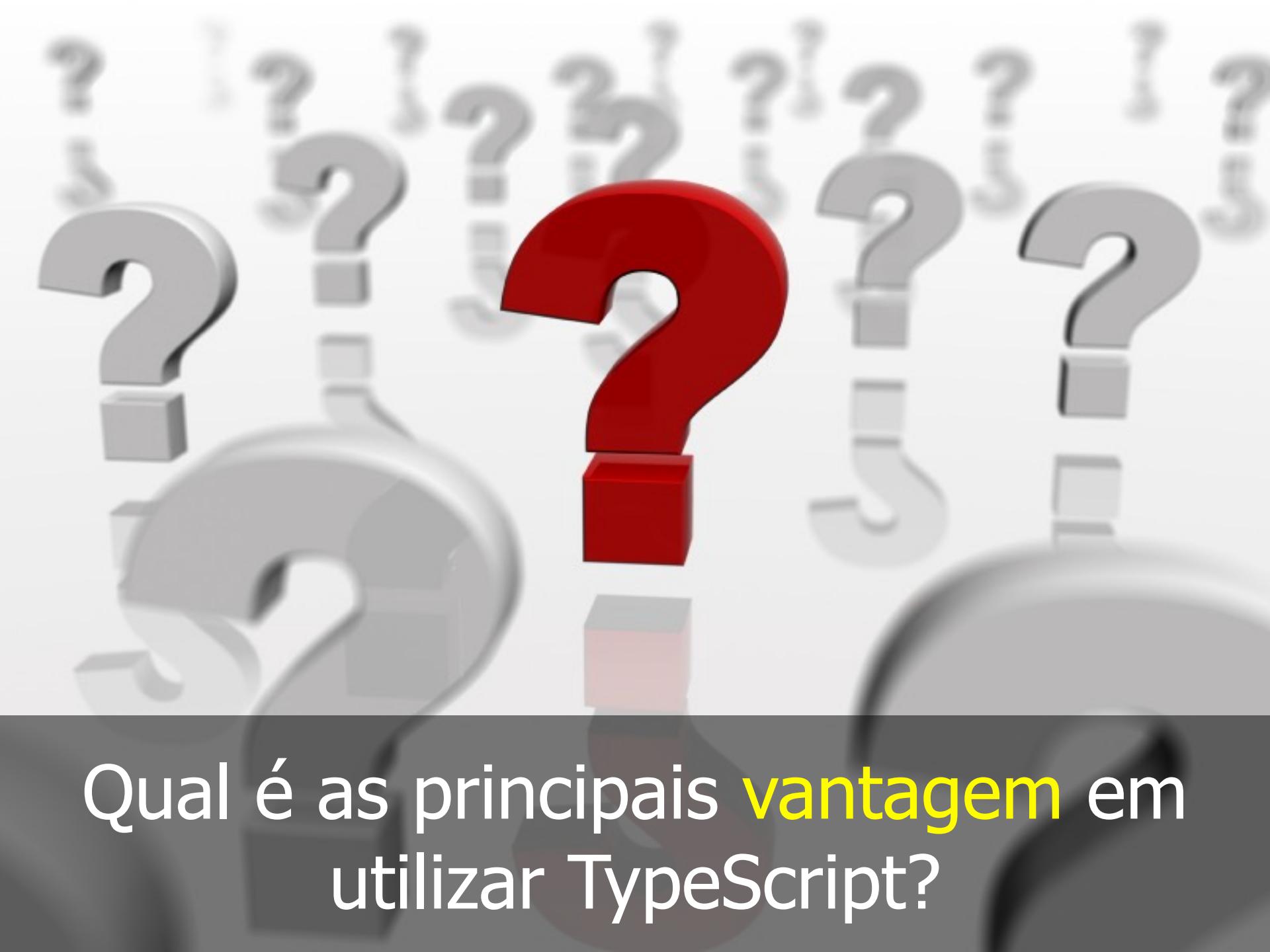




TypeScript



Qual é as principais vantagem em
utilizar TypeScript?

Pense nos **erros** que existem na linguagem JavaScript, em tempo de interpretação. Boa parte deles seria facilmente evitada se uma verificação da tipagem fosse realizada enquanto programamos

main.ts — typescript

```
TS main.ts 1 ●
```

```
1 const message = "Hello World";
2
3 message();
4
```

PROBLEMS 1 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) Y ≡ ^ X

TS main.ts src 1

ⓘ ^ This expression is not callable.
Type 'String' has no call signatures. ts(2349) [3, 1]

main.ts — typescript

```
TS main.ts 1 ●
```

```
1 const user = {
2   name: "Rodrigo Branas",
3   email: "rodrigo@branas.io"
4 };
5
6 user.city = "Florianópolis";
7
```

PROBLEMS 1 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) Y ≡ ^ X

TS main.ts src 1

✖ Property 'city' does not exist on type '{ name: string; email: string; }'. ts(2339) [6, 6]

main.ts — typescript

```
TS main.ts 1 ●
```

```
1 const factor = 10;
2
3 factor.toFixed(2);
4
```

PROBLEMS 1 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) Y ≡ ^ X

TS main.ts src 1

> 💡 Property 'toFixed' does not exist on type '10'. Did you mean 'toFixed'? ts(2551) [3, 8]

Além disso, temos mais suporte a OO, permitindo uma implementação muito mais alinhada com os princípios do SOLID e especialmente em projetos com regras de negócio mais complexas

Não que não seja possível **utilizar abordagens orientadas ao domínio** como o DDD com JavaScript mas tipos, interfaces, classes abstratas e modificadores de visibilidade fazem muita diferença na implementação

Existem ainda funcionalidades como os **decorators**, que ajudam muito na configuração de frameworks e bibliotecas

Por fim, antecipar as novas funcionalidades que ainda estão em stage 3 e 4 no TC39 e que em breve serão lançadas na linguagem JavaScript

tc39/proposals: Tracking ECMAScript proposals

github.com/tc39/proposals

Search or jump to... / Pull requests Issues Marketplace Explore

tc39 / proposals Public Watch 2.2k Fork 688 Star 15.6k

Code Issues 3 Pull requests 3 Actions Security Insights

main 3 branches 0 tags Go to file Add file Code

dnlborczyk and ljharb add missing tests column cell cc332b6 6 days ago 824 commits

ecma402 Add proposal-intl-temporal 13 days ago

CODE_OF_CONDUCT.md fix links 2 years ago

CONTRIBUTING.md [meta] link to HEAD instead of master in github links 14 months ago

ISSUE_TEMPLATE.md [meta] link to HEAD instead of master in github links 14 months ago

README.md add missing tests column cell 6 days ago

finished-proposals.md Array find from last to stage 4, per 2022.06.06 TC39 23 days ago

inactive-proposals.md fix: links 4 months ago

stage-0-proposals.md move deprecated proposal to inactive 11 months ago

stage-1-proposals.md Update the last meeting note for Pattern Matching proposal 18 days ago

README.md

ECMAScript proposals

- Stage 1 Proposals
- Stage 0 Proposals
- Finished Proposals
- Inactive Proposals

ECMAScript Internationalization API Specification proposals

Contributing new proposals

About

Tracking ECMAScript Proposals

tc39.github.io/process-document/

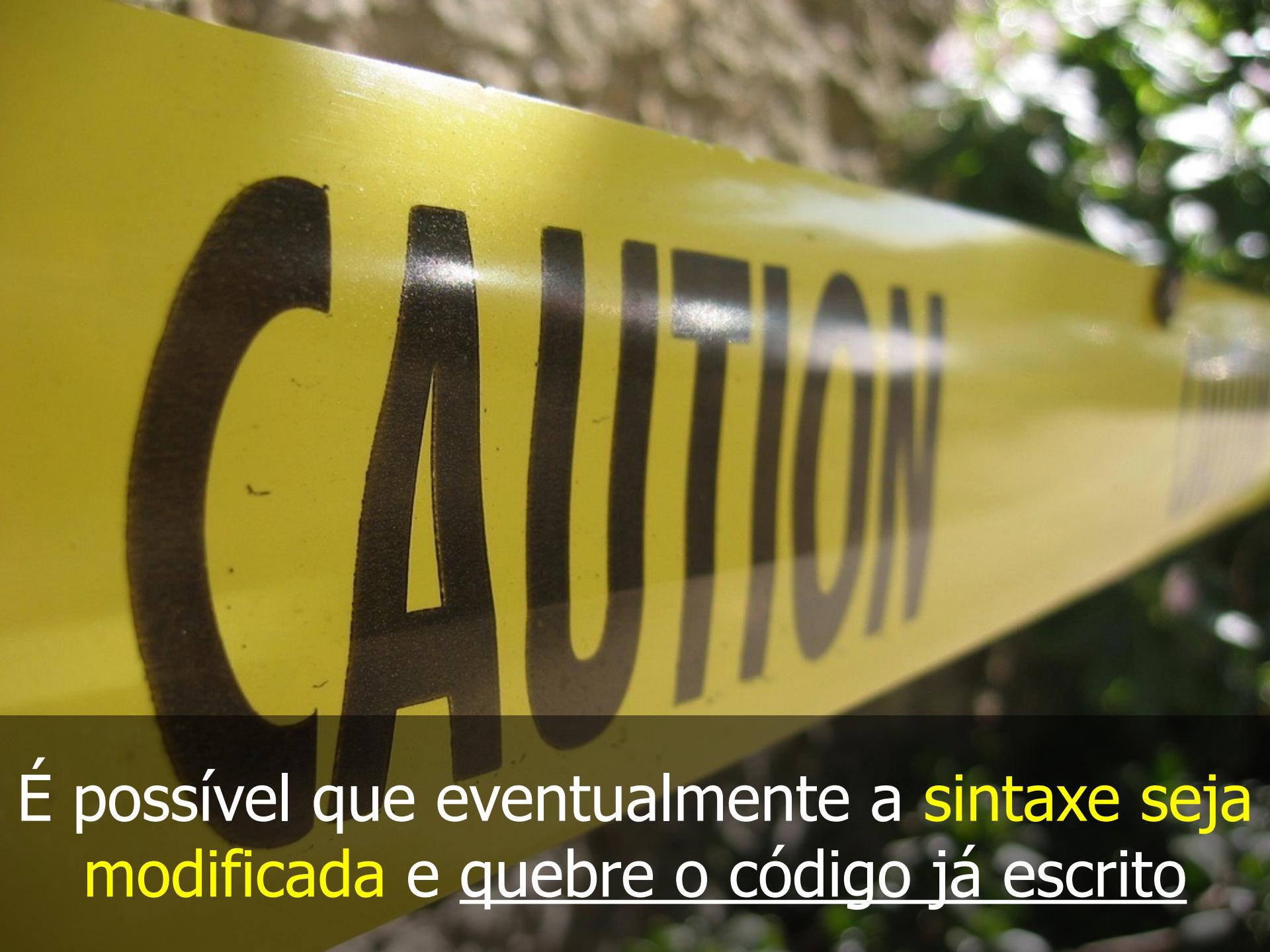
javascript language proposal
ecmascript spec specification
language-design committee

Readme Code of conduct 15.6k stars 2.2k watching 688 forks

Contributors 101 + 90 contributors

Active proposals				
<hr/>				
Proposal	Author	Champion	Tests	Last Presented
Legacy RegExp features in JavaScript	Claude Pache	Mark Miller Claude Pache	<input checked="" type="checkbox"/>	May 2017
Hashbang Grammar	Bradley Farias	Bradley Farias	<input checked="" type="checkbox"/>	November 2018
Atomics.waitAsync	Lars Hansen	Shu-yu Guo Lars Hansen	<input checked="" type="checkbox"/>	December 2019
Import Assertions	Myles Borins Sven Sauleau Dan Clark Daniel Ehrenberg	Myles Borins Sven Sauleau Dan Clark Daniel Ehrenberg	<input checked="" type="checkbox"/>	November 2020
JSON Modules	Myles Borins Sven Sauleau Dan Clark Daniel Ehrenberg	Myles Borins Sven Sauleau Dan Clark Daniel Ehrenberg	<input checked="" type="checkbox"/>	January 2021
Temporal	Philipp Dunkel Maggie Johnson-Pint Matt Johnson-Pint Brian Terlson Shane Carr Ujjwal Sharma Philip Chimento Jason Williams Justin Grant	Philipp Dunkel Maggie Johnson-Pint Matt Johnson-Pint Brian Terlson Shane Carr Ujjwal Sharma Philip Chimento Jason Williams Justin Grant	<input checked="" type="checkbox"/>	March 2021
Resizable and growable ArrayBuffer	Shu-yu Guo	Shu-yu Guo	<input checked="" type="checkbox"/>	May 2021

README.md			
Stage 2			
Proposal	Author	Champion	Last Presented
function.sent metapropery	Allen Wirfs-Brock	HE Shi-Jun	July 2019
throw expressions	Ron Buckton	Ron Buckton	January 2018
Function implementation hiding	Domenic Denicola Michael Ficarra	Michael Ficarra	June 2020
New Set methods	Michał Wadas Sathya Gunasekaran	Sathya Gunasekaran	January 2019
Sequence properties in Unicode property escapes	Mathias Bynens	Mathias Bynens	October 2019
collection normalization	Bradley Farias	Bradley Farias	January 2019
Array.isTemplateObject	Mike Samuel, Krzysztof Kotowicz	Krzysztof Kotowicz	December 2019
Iterator helpers	Gus Caplan	Michael Ficarra Jonathan Keslin	July 2020
Explicit Resource Management	Ron Buckton	Ron Buckton	February 2020
Map.prototype.emplace	Bradley Farias	Erica Pramer	July 2020
Dynamic Import Host Adjustment	Mike Samuel, Krzysztof Kotowicz	Krzysztof Kotowicz	December 2019
WeakRefs cleanupSome	Yulia Startsev Daniel Ehrenberg	Yulia Startsev Daniel Ehrenberg	July 2020
Record & Tuple	Robin Ricard Richard Button	Robin Ricard Richard Button	December 2021
JSON.parse source text access	Richard Gibson	Richard Gibson	July 2020
Module Blocks	Surma Daniel Ehrenberg	Surma	January 2021
Pipeline Operator	J. S. Choi James DiGioia Ron Buckton	J. S. Choi Ron Buckton	August 2021

A close-up photograph of a yellow caution tape. The word "CAUTION" is printed in large, bold, black capital letters. The tape is slightly curved, and the background shows some green foliage.

CAUTION

É possível que eventualmente a sintaxe seja
modificada e quebre o código já escrito



O que é e para que serve o arquivo
tsconfig.json?

O arquivo tsconfig.json centraliza todas as regras de análise estática e de transpilação e é importante configurá-lo de forma adequada, conforme as necessidades e políticas do projeto

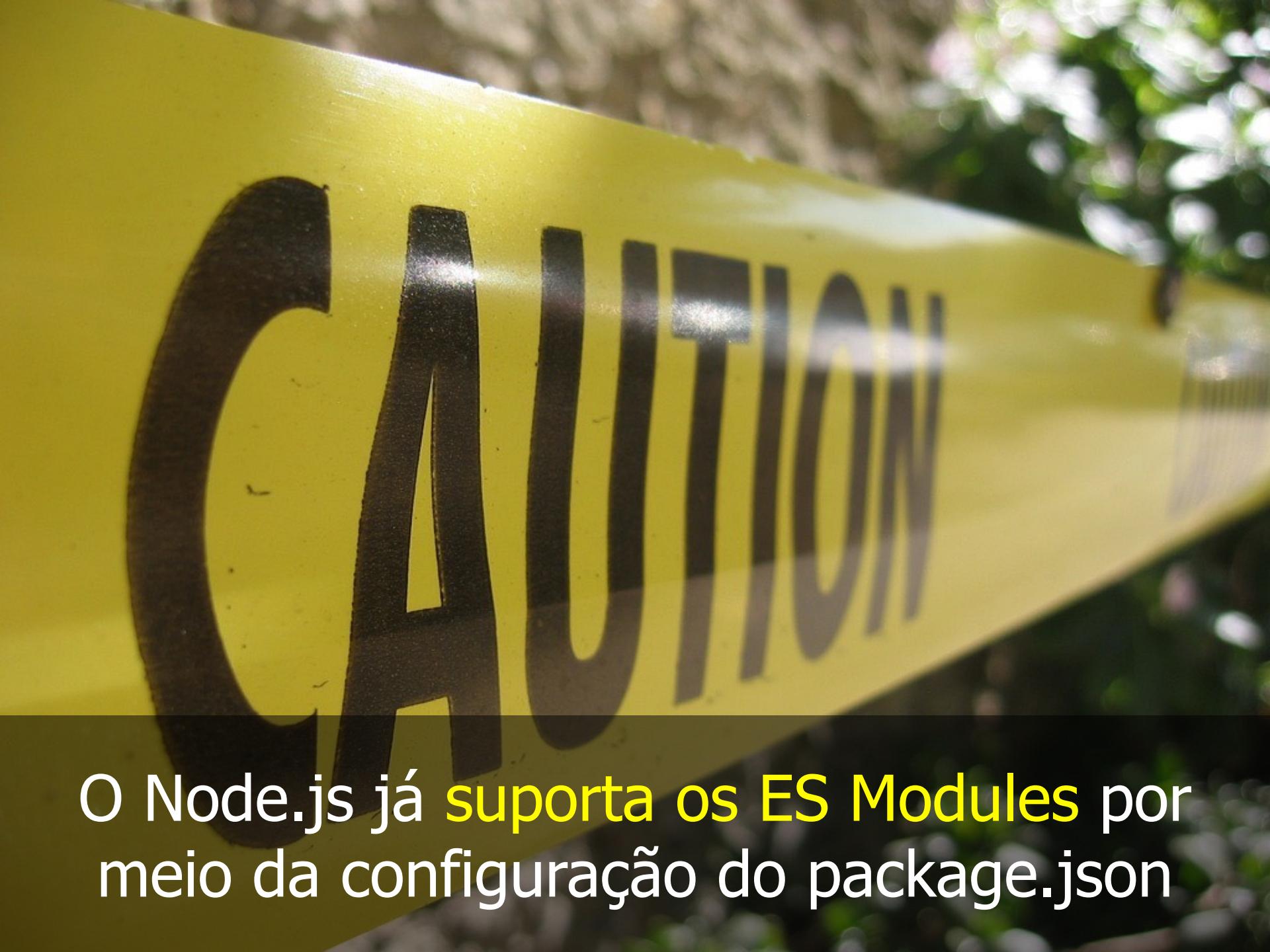
tsconfig.json — typescript

```
1  {
2    "compilerOptions": {
3      /* Visit https://aka.ms/tsconfig.json to read more about this file */
4
5      /* Projects */
6      // "incremental": true,                                     /* Enable incremental compilation */
7      // "composite": true,                                     /* Enable constraints that allow a TypeScript pro */
8      // "tsBuildInfoFile": "./",                                /* Specify the folder for .tsbuildinfo incrementa */
9      // "disableSourceOfProjectReferenceRedirect": true,        /* Disable preferring source files instead of dec */
10     // "disableSolutionSearching": true,                      /* Opt a project out of multi-project reference c */
11     // "disableReferencedProjectLoad": true,                   /* Reduce the number of projects loaded automatic */
12
13     /* Language and Environment */
14     "target": "es2016",                                       /* Set the JavaScript language version for emitte */
15     // "lib": [],                                            /* Specify a set of bundled library declaration f */
16     // "jsx": "preserve",                                     /* Specify what JSX code is generated. */
17     // "experimentalDecorators": true,                        /* Enable experimental support for TC39 stage 2 d */
18     // "emitDecoratorMetadata": true,                          /* Emit design-type metadata for decorated declar */
19     // "jsxFactory": "",                                    /* Specify the JSX factory function used when tar */
20     // "jsxFragmentFactory": "",                            /* Specify the JSX Fragment reference used for fr */
21     // "jsxImportSource": "",                               /* Specify module specifier used to import the JS */
22     // "reactNamespace": "",                               /* Specify the object invoked for `createElement` */
23     // "noLib": true,                                         /* Disable including any library files, including */
24     // "useDefineForClassFields": true,                      /* Emit ECMAScript-standard-compliant class field */
25
26     /* ... */
27   }
28 }
```

A opção **incremental** serve para habilitar a transpilação incremental, guardando uma cache que agiliza futuras transpilações especialmente em projetos muito grandes

A opção **target** define qual é a versão da linguagem JavaScript que será utilizada como destino da transpilação

A opção **module** é importante para definir o tipo de sistema de módulo que será utilizado, sendo que os mais importantes são CommonJS ou os ES Modules, lançado no ES6

A close-up photograph of a yellow caution tape. The word "CAUTION" is printed in large, bold, black capital letters. The tape is slightly curved, and the background shows some green foliage.

CAUTION

O Node.js já suporta os **ES Modules** por meio da configuração do package.json

A opção **include** define quais são os diretórios que devem ser transpilados pelo TypeScript enquanto o **exclude** ignora um conjunto de diretórios

Emit

O grupo de configurações emit serve para definir parâmetros relacionados com a transpilação

A opção **outFile** é muito utilizada para projetos que rodam no browser definindo um arquivo que vai concatenar todos os arquivos que foram transpilados

A opção **outDir** é essencial já que define o diretório de armazenamento dos arquivos transpilados

Definição de tipos

Definição dos **tipos** em variáveis, parâmetros e retornos de funções e propriedades de objetos

main.ts — typescript

TS main.ts X

```
1 const email: string = "rodrigo@branas.io";
2 |
```

PROBLEMS OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) Y X

No problems have been detected in the workspace.

Tipos em variáveis

The screenshot shows a code editor window titled "main.ts — typescript". The code in the editor is:

```
1 let email: string;
2 email = 10;
```

The line "email = 10;" has a red wavy underline under the number "10", indicating a type error. The editor interface includes a "PROBLEMS" tab with a count of 1, a "Filter" bar, and a "OUTPUT" tab. The problem list shows:

- TS main.ts src 1
⊗ Type 'number' is not assignable to type 'string'. ts(2322) [Ln 2, Col 1]

Não é possível atribuir um valor do tipo errado na variável

main.ts — typescript

```
1 const email = "rodrigo@branas.io";
2
```

PROBLEMS OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) X

No problems have been detected in the workspace.

Ao criar uma variável e atribuir um valor o
TS infere o tipo automaticamente

The screenshot shows a Microsoft Visual Studio Code interface. The top bar indicates the file is named "main.ts — typescript". The main editor area contains the following TypeScript code:

```
1  function init (port: number) {  
2      console.log(port);  
3  }  
4  
5  init(80);  
6
```

The number 80 in the call to the `init` function is highlighted in green, indicating it is a literal value. The code editor has a light gray theme.

Below the editor is a navigation bar with tabs: PROBLEMS (underlined), OUTPUT, and To the right of the tabs is a search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)" with a magnifying glass icon. Further right are icons for a file, a refresh arrow, and a close button.

The message "No problems have been detected in the workspace." is displayed below the filter bar.

Tipos em parâmetros

The screenshot shows a code editor window for a file named `main.ts` with the extension `-typescript`. The code contains a function `init` that logs its parameter `port` to the console. The invocation of `init` with the argument `"80"` is highlighted with a yellow background, indicating a TypeScript type error. A small yellow lightbulb icon is positioned next to the number 4, suggesting a potential fix or refactoring suggestion.

```
TS main.ts 1 ●  
1   function init (port: number) {  
2     console.log(port);  
3   }  
4   init("80");  
5  
6
```

Below the editor, the VS Code interface includes a **PROBLEMS** tab with 1 error, an **OUTPUT** tab, and a **...** tab. A search bar labeled **Filter (e.g. text, **/*.ts, !**/node_modules/**)** is also present. The error message in the problems list is:

TS main.ts src 1
× Argument of type 'string' is not assignable to parameter of type 'number'. ts(2345) [5, 6]

Não é possível passar um parâmetro com o tipo errado ao invocar a função

The screenshot shows a code editor window titled "main.ts – typescript". The code in the editor is:

```
1  function add (a: number, b: number): number {  
2      return a + b;  
3  }  
4  
5  const result = add(2, 2);  
6  console.log(result);  
7
```

The code defines a function named "add" that takes two numbers as parameters and returns their sum. It then creates a variable "result" by calling "add" with arguments 2 and 2, and logs "result" to the console.

Below the editor, there is a "PROBLEMS" tab, an "OUTPUT" tab, and a "Filter (e.g. text, **/*.ts, !**/node_modules/**)" input field. The "PROBLEMS" tab is underlined, indicating it is active. The message "No problems have been detected in the workspace." is displayed.

Tipos em retornos de função

The screenshot shows a code editor window for a file named `main.ts`. The code contains a function `add` that returns a string concatenation of two numbers. A red squiggly underline is under the return statement, indicating a type error. The code is as follows:

```
1  function add (a: number, b: number): number {  
2      return `${a + b}`;  
3  }  
4  
5  const result = add(2, 2);  
6  console.log(result);  
7
```

Below the editor, the `PROBLEMS` tab is selected, showing one error: `Type 'string' is not assignable to type 'number'. ts(2322) [2, 2]`.

Não é possível retornar um valor com o tipo errado

The screenshot shows a code editor window titled "main.ts – typescript". The code in the editor is:

```
1  function add (a: number, b: number) {
2      return a + b;
3  }
4
5  const result = add(2, 2);
6  console.log(result);
7
```

The code consists of a single function named "add" that takes two numbers as parameters and returns their sum. It is then called with arguments 2 and 2, and the result is logged to the console.

Below the editor, the "PROBLEMS" tab is selected, showing the message: "No problems have been detected in the workspace."

O tipo de retorno é **inferido**
automaticamente

The screenshot shows a code editor window titled "main.ts – typescript". The code in the editor is:

```
TS main.ts 1 ×  
1 function add (a: number, b: number) {  
2     return `${a + b}`;  
3 }  
4  
5 const result: number = add(2, 2);  
6 console.log(result);  
7
```

The variable declaration "const result: number = add(2, 2);" is highlighted with a yellow background, indicating a potential issue. The word "result" is underlined with a red wavy line, indicating a spelling error or a reference to an undeclared variable.

Below the editor, the "PROBLEMS" tab is selected, showing one error: "Type 'string' is not assignable to type 'number'. ts(2322) [5, 7]".

Não é possível atribuir o retorno de
uma função ao tipo errado

The screenshot shows a code editor window titled "main.ts — typescript". The code is written in TypeScript and defines a Person class with properties name and age, and a constructor that initializes them. A variable person is then created and assigned an object with the name "Rodrigo Branas" and age 38.

```
1  class Person {  
2      name: string;  
3      age: number;  
4  
5      constructor (name: string, age: number) {  
6          this.name = name;  
7          this.age = age;  
8      }  
9  }  
10 const person = new Person("Rodrigo Branas", 38);  
11
```

Below the code editor is a "PROBLEMS" tab, which is currently active, followed by "OUTPUT" and "...". To the right of the tabs is a search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)" with a magnifying glass icon. Further to the right are icons for refresh, close, and other navigation functions.

No problems have been detected in the workspace.

Tipos nas propriedades de uma classe

The screenshot shows a code editor window titled "main.ts — typescript". The code defines a Person class with a constructor that expects a string for name and a number for age. A line of code creates a person instance with a string name and a boolean value for age, which is highlighted in yellow and underlined with red, indicating a TypeScript error.

```
1  class Person {  
2      name: string;  
3      age: number;  
4  
5      constructor (name: string, age: number) {  
6          this.name = name;  
7          this.age = age;  
8      }  
9  
10     const person = new Person("Rodrigo Branas", true);  
11
```

Below the editor, the "PROBLEMS" tab is active, showing one error: "Argument of type 'boolean' is not assignable to parameter of type 'number'. ts(2345) [10, 45]".

Não é possível construir a instância
passando um parâmetro do tipo errado

A screenshot of a code editor window titled "main.ts — typescript". The code editor displays the following TypeScript code:

```
1 const numbers: number[] = [];
2
3 numbers[0] = 1;
4 numbers[1] = 2;
5 numbers[2] = 3;
6
```

The code defines a constant `numbers` of type `number[]` and initializes it with three elements: 1, 2, and 3.

Below the code editor is a "PROBLEMS" tab, which is currently active, showing the message "No problems have been detected in the workspace."

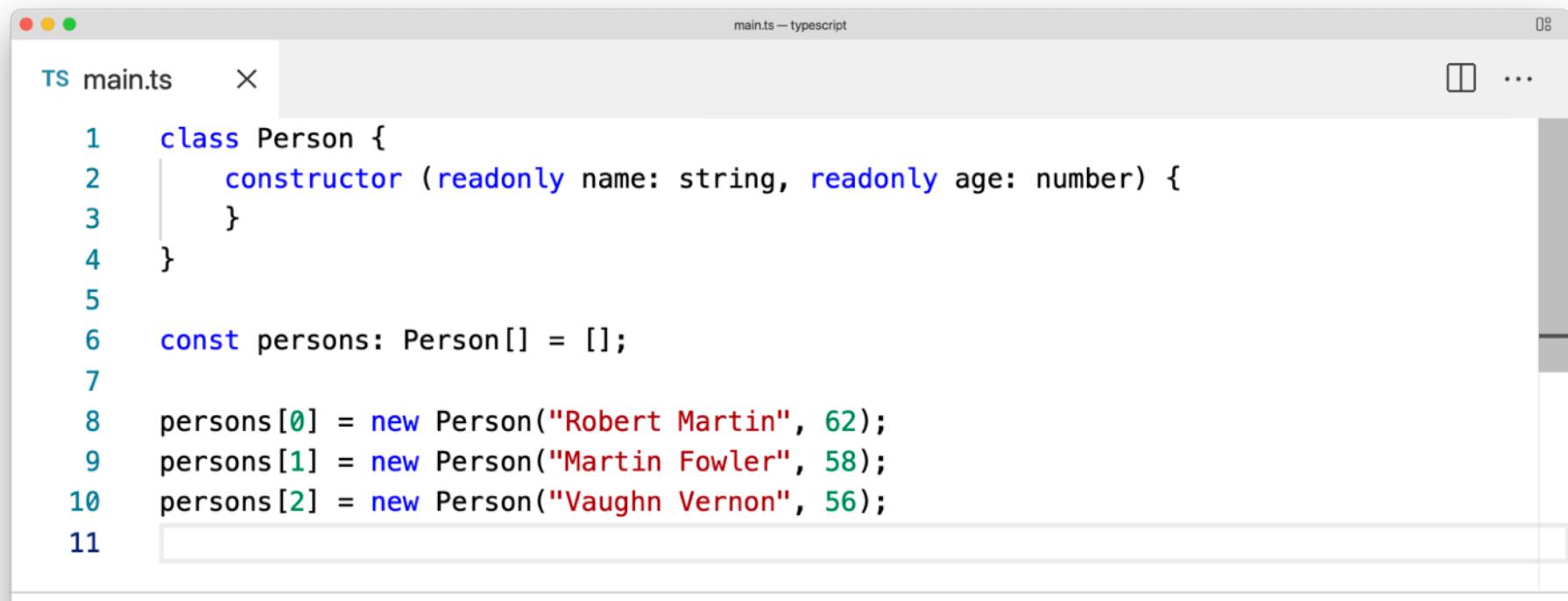
Tipos em um **array**

The screenshot shows a code editor window titled "main.ts — typescript". The code in the editor is:

```
1 const numbers: number[] = [];
2
3 numbers[0] = 1;
4 numbers[1] = 2;
5 numbers[2] = "3";
6
```

The line "numbers[2] = "3"" is highlighted with a yellow background, indicating a potential issue. In the bottom-left corner, there is a "PROBLEMS" tab with a count of 1, and below it, a list item for "main.ts" with a count of 1. The list item is expanded, showing a red circular icon with an "X" and the message: "Type 'string' is not assignable to type 'number'. ts(2322) [5, 1]".

Não é possível inserir elementos com tipos errados



A screenshot of a code editor window titled "main.ts — typescript". The code defines a "Person" class with a constructor taking "name" and "age" as readonly properties. An array "persons" is created and populated with three instances of "Person" for "Robert Martin", "Martin Fowler", and "Vaughn Vernon". The code editor interface includes tabs for "PROBLEMS", "OUTPUT", and "...", and a "Filter" bar at the bottom.

```
1  class Person {  
2      constructor (readonly name: string, readonly age: number) {  
3      }  
4  }  
5  
6  const persons: Person[] = [];  
7  
8  persons[0] = new Person("Robert Martin", 62);  
9  persons[1] = new Person("Martin Fowler", 58);  
10 persons[2] = new Person("Vaughn Vernon", 56);  
11
```

No problems have been detected in the workspace.

É possível definir **classes** também,
como tipos de um array

The screenshot shows a code editor window titled "main.ts — typescript". The code defines a "Person" class with a constructor taking "name" and "age" as readonly properties. An array "persons" is created and populated with three elements: a new instance of Person with name "Robert Martin" and age 62, a new instance with name "Martin Fowler" and age 58, and an object { name: "Vaughn Vernon", age: 56 }.

```
1  class Person {
2      constructor (readonly name: string, readonly age: number) {
3      }
4  }
5
6  const persons: Person[] = [];
7
8  persons[0] = new Person("Robert Martin", 62);
9  persons[1] = new Person("Martin Fowler", 58);
10 persons[2] = { name: "Vaughn Vernon", age: 56 };
11
```

Below the editor is a "PROBLEMS" tab bar with "OUTPUT" and "...". A search bar says "Filter (e.g. text, **/*.ts, !**/node_modules/**)". The message "No problems have been detected in the workspace." is displayed.

Ao definir classes como tipos, é inserir
objetos compatíveis no array

```
TS main.ts 1 ● main.ts — typescript

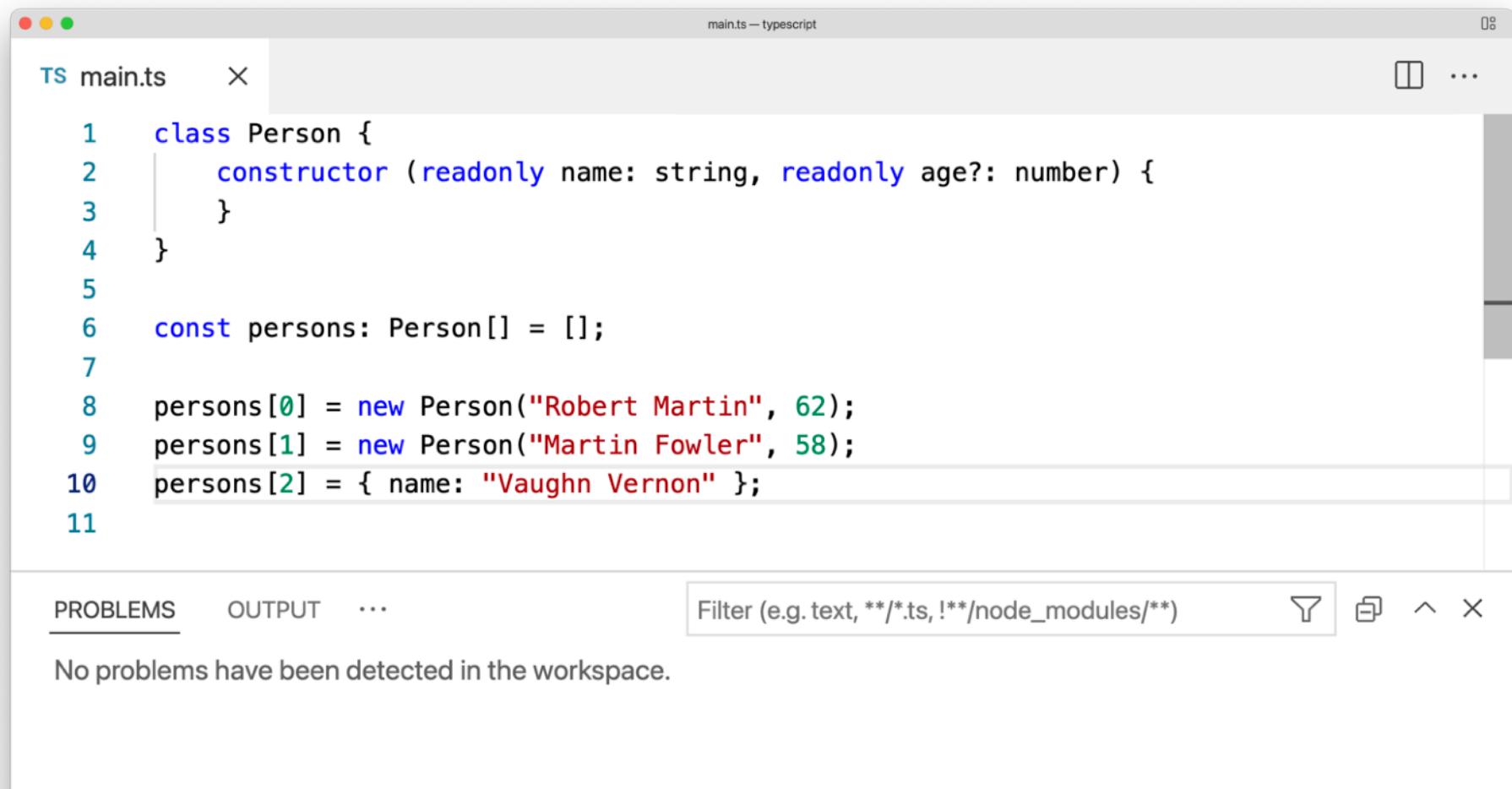
1  class Person {
2      constructor (readonly name: string, readonly age: number) {
3      }
4  }
5
6  const persons: Person[] = [];
7
8  persons[0] = new Person("Robert Martin", 62);
9  persons[1] = new Person("Martin Fowler", 58);
10 persons[2] = { name: "Vaughn Vernon" };
11
```

PROBLEMS 1 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

TS main.ts src 1

> ✘ Property 'age' is missing in type '{ name: string; }' but required in type 'Person'. ts(2741) [10, 1]

Não é possível inserir objetos
incompatíveis com a classe

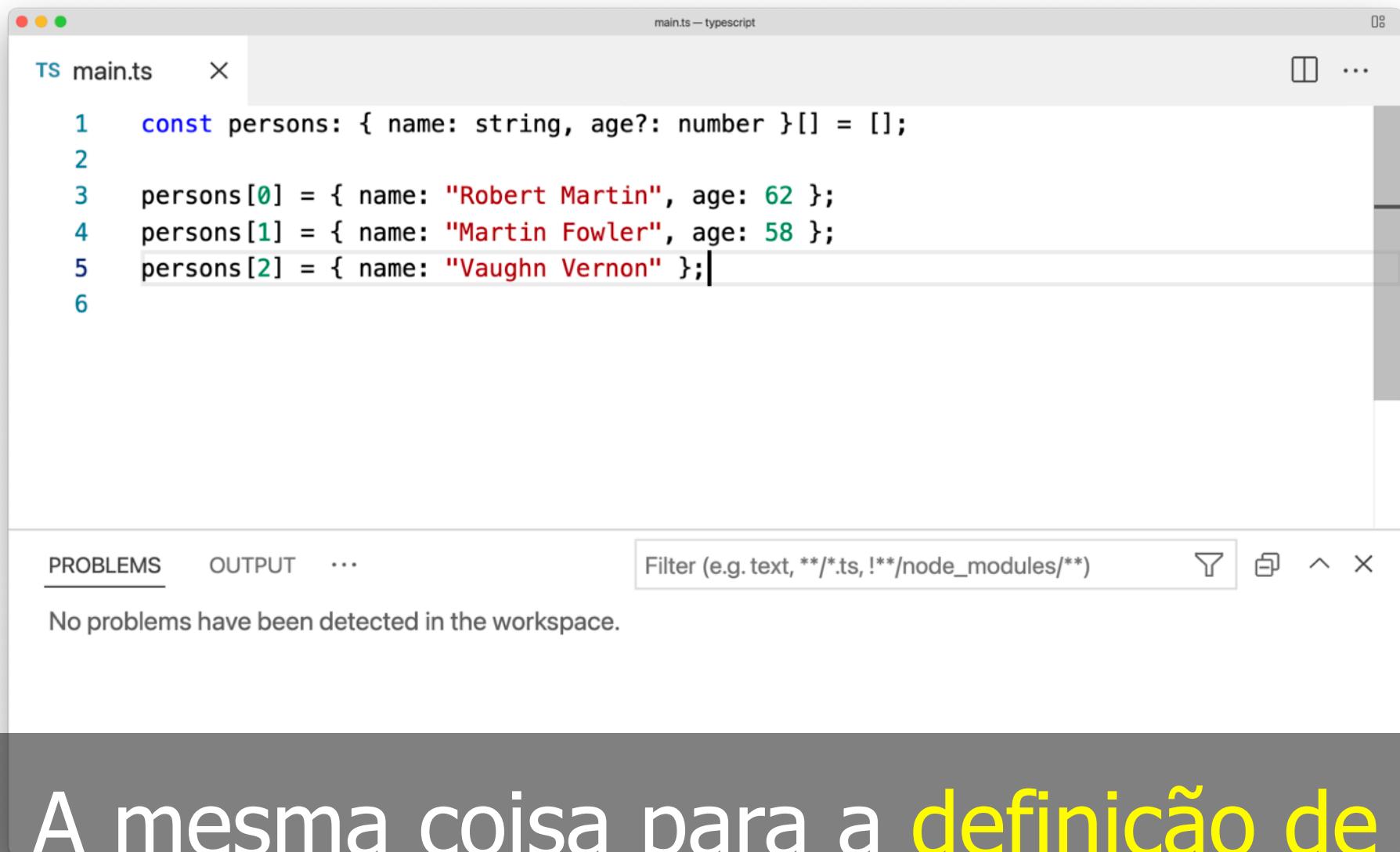


The screenshot shows a code editor window titled "main.ts — typescript". The code defines a Person class with a constructor that takes a required name string and an optional age number. An array of Person objects is created, with the third element being an object literal instead of a new instance.

```
1  class Person {  
2      constructor (readonly name: string, readonly age?: number) {  
3      }  
4  }  
5  
6  const persons: Person[] = [];  
7  
8  persons[0] = new Person("Robert Martin", 62);  
9  persons[1] = new Person("Martin Fowler", 58);  
10 persons[2] = { name: "Vaughn Vernon" };  
11
```

Below the code editor is a "PROBLEMS" tab, which is currently selected. The output area below it displays the message: "No problems have been detected in the workspace."

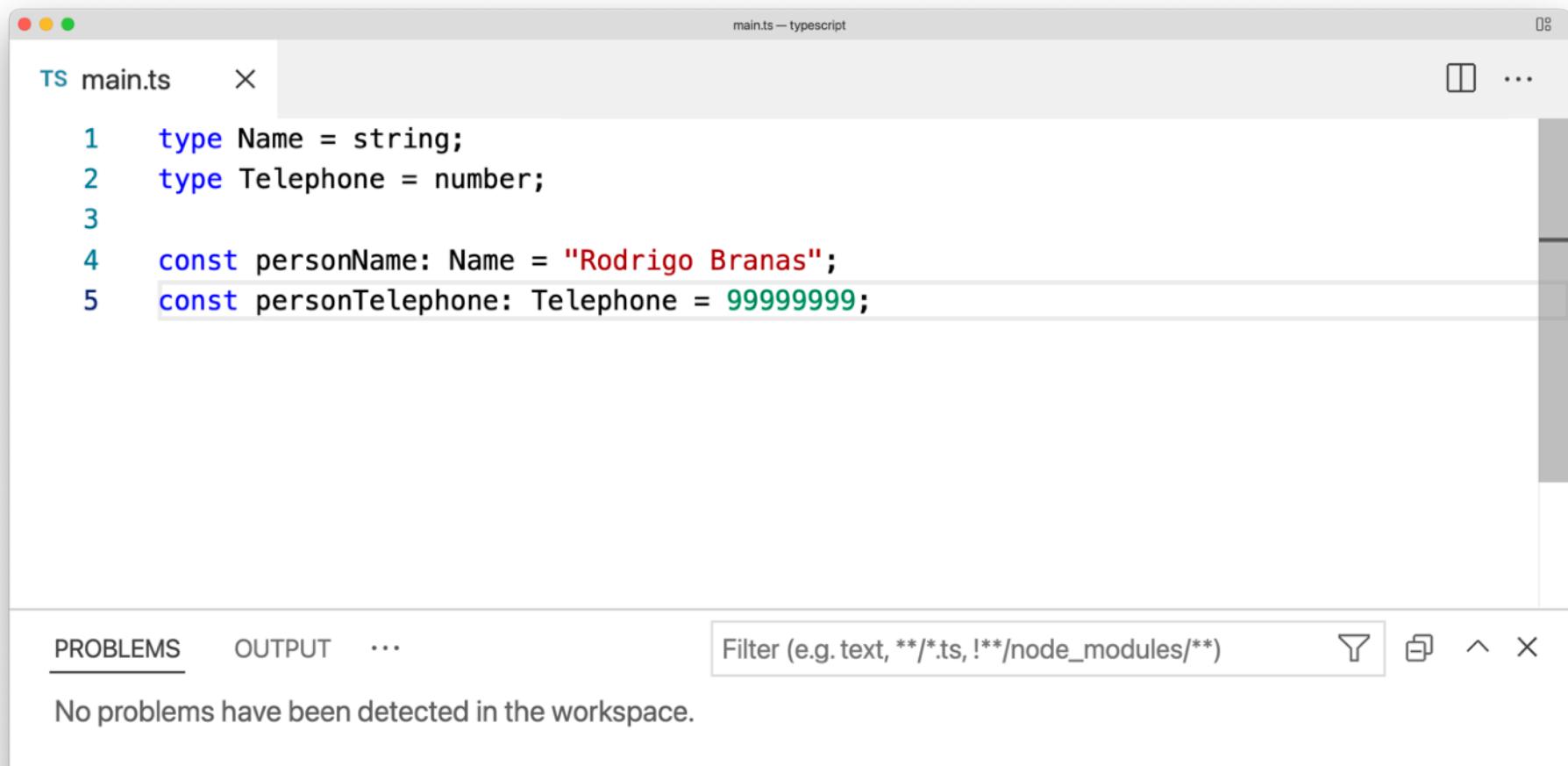
É possível utilizar a ? para definir uma propriedade como opcional



A screenshot of a code editor window titled "main.ts — typescript". The code defines an array of objects named "persons" with properties "name" (string) and "age" (number). The array has three elements, indexed from 0 to 2. The first two elements have their "name" and "age" properties explicitly set, while the third element only has its "name" property set. The code editor interface includes tabs for "PROBLEMS", "OUTPUT", and "...". Below the tabs, a message states "No problems have been detected in the workspace."

```
1  const persons: { name: string, age?: number }[] = [];
2
3  persons[0] = { name: "Robert Martin", age: 62 };
4  persons[1] = { name: "Martin Fowler", age: 58 };
5  persons[2] = { name: "Vaughn Vernon" };|
6
```

A mesma coisa para a definição de tipos



A screenshot of a code editor window titled "main.ts — typescript". The code editor displays the following TypeScript code:

```
1  type Name = string;
2  type Telephone = number;
3
4  const personName: Name = "Rodrigo Branas";
5  const personTelephone: Telephone = 99999999;
```

The code editor interface includes a toolbar at the top with icons for file operations, and a bottom bar with tabs for "PROBLEMS", "OUTPUT", and "...". The "PROBLEMS" tab is active, showing the message: "No problems have been detected in the workspace."

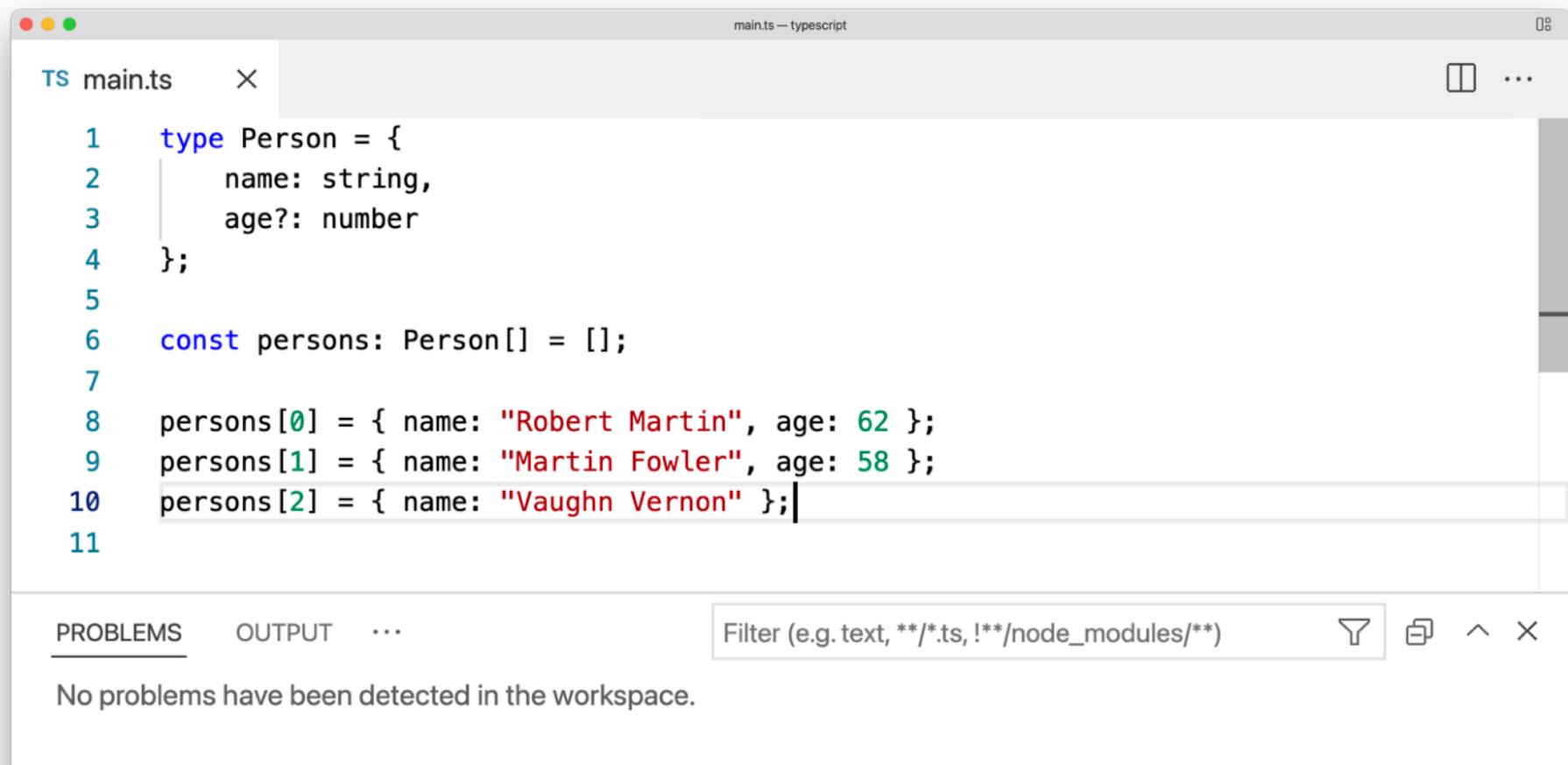
A definição de tipos, ou Type Alias,
permite reusá-los

The screenshot shows a code editor window titled "main.ts — typescript". The code defines two types: "Name" as a string and "Telephone" as a number. It then creates two constants: "personName" with the value "Rodrigo Branas" and "personTelephone" with the value "+55 11 9999-9999". The "personTelephone" assignment is highlighted with a red underline, indicating a TypeScript error.

```
1 type Name = string;
2 type Telephone = number;
3
4 const personName: Name = "Rodrigo Branas";
5 const personTelephone: Telephone = "+55 11 9999-9999";
```

Below the editor, the "PROBLEMS" tab is active, showing one error: "Type 'string' is not assignable to type 'number'. ts(2322) [5, 7]".

Não é possível atribuir um valor do tipo errado a uma variável tipada



The screenshot shows a code editor window titled "main.ts — typescript". The code defines a "Person" type with properties "name" (string) and "age?" (number). An array "persons" is created and populated with three objects, each having a "name" and an "age". The "age" property is marked as optional (indicated by a question mark) in the type definition.

```
1  type Person = {
2      name: string,
3      age?: number
4  };
5
6  const persons: Person[] = [];
7
8  persons[0] = { name: "Robert Martin", age: 62 };
9  persons[1] = { name: "Martin Fowler", age: 58 };
10 persons[2] = { name: "Vaughn Vernon" };|
```

Below the editor is a "PROBLEMS" tab, which is currently active, showing that no problems have been detected. There is also an "OUTPUT" tab and a "Filter" search bar.

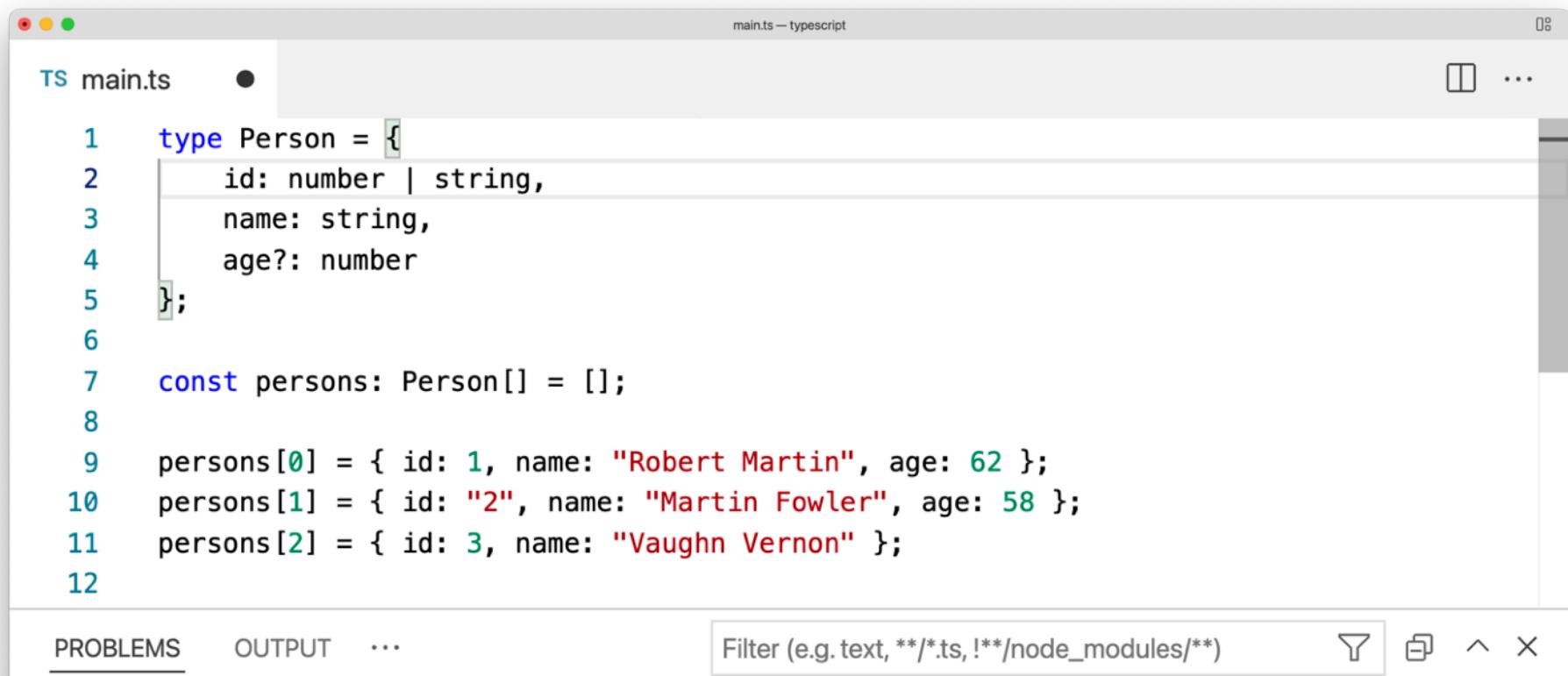
É possível definir **propriedades opcionais** nos tipos

The screenshot shows a code editor window titled "main.ts — typescript". The code defines a Person type and initializes an array of persons. The problematic part is at line 9, where the first element's id is assigned a string value ("1") instead of a number. This causes a TypeScript error: "Type 'string' is not assignable to type 'number'".

```
1  type Person = {
2    id: number,
3    name: string,
4    age?: number
5  };
6
7  const persons: Person[] = [];
8
9  persons[0] = { id: 1, name: "Robert Martin", age: 62 };
10 persons[1] = { id: "2", name: "Martin Fowler", age: 58 };
11 persons[2] = { id: 3, name: "Vaughn Vernon" };
12
```

Below the code editor, the "PROBLEMS" tab is active, showing one error. The error message is: "Type 'string' is not assignable to type 'number'. ts(2322) [10,16]".

Não é inserir elemtnos com
propriedades do tipo errado em arrays



The screenshot shows a code editor window titled "main.ts — typescript". The code defines a "Person" type with properties: id (number or string), name (string), and age? (number). It then creates an array of three Person objects: Robert Martin (id 1, age 62), Martin Fowler (id "2", age 58), and Vaughn Vernon (id 3, age null).

```
1  type Person = {
2      id: number | string,
3      name: string,
4      age?: number
5  };
6
7  const persons: Person[] = [];
8
9  persons[0] = { id: 1, name: "Robert Martin", age: 62 };
10 persons[1] = { id: "2", name: "Martin Fowler", age: 58 };
11 persons[2] = { id: 3, name: "Vaughn Vernon" };
12
```

PROBLEMS OUTPUT ...

Filter (e.g. text, **/*.ts, !**/node_modules/**)



No problems have been detected in the workspace.

É possível definir mais de um tipo
utilizando Union Types

A screenshot of a TypeScript code editor window titled "main.ts – typescript". The code defines three types: Entity, Person, and PersonEntity, and creates a constant personEntity of type PersonEntity.

```
TS main.ts ×
1  type Entity = {
2    | id: number | string
3  };
4
5  type Person = {
6    name: string,
7    age: number
8  };
9
10 type PersonEntity = Entity & Person;
11
12 const personEntity: PersonEntity = { id: 1, name: "Robert Martin", age: 62};
```

The code editor interface includes tabs for PROBLEMS, OUTPUT, and ... at the bottom left, and a Filter bar at the bottom right. The status bar at the bottom shows "No problems have been detected in the workspace."

Também é possível juntar tipos com um Merge Type

The screenshot shows a VS Code window with the following code in `main.ts`:

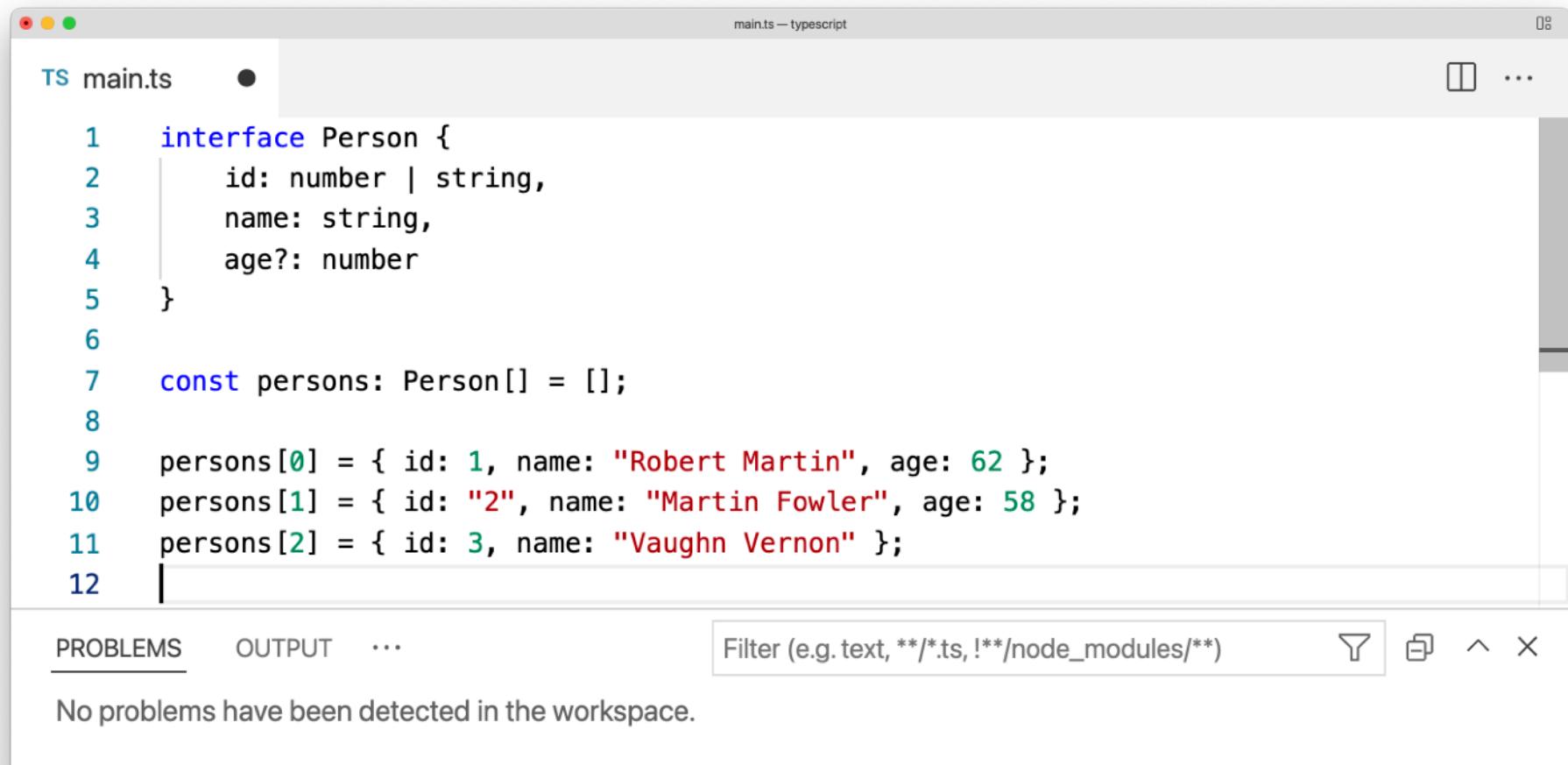
```
TS main.ts 1 ×
1  type Entity = {
2  |   id: number | string
3  };
4
5  type Person = {
6  |   name: string,
7  |   age: number
8  };
9
10 type PersonEntity = Entity & Person;
11
12 const personEntity: PersonEntity = { name: "Robert Martin", age: 62};
```

The variable `personEntity` is highlighted in yellow. The line `const personEntity: PersonEntity = { name: "Robert Martin", age: 62};` is underlined with red, indicating a TypeScript error.

In the bottom left, the 'PROBLEMS' tab has 1 error. The error message is:

> ✖ ^ Type '{ name: string; age: number; }' is not assignable to type 'PersonEntity'.
Property 'id' is missing in type '{ name: string; age: number; }' but required in type 'Entity'. ts(2322) [12, 7]

Os Merge Types são verificados e não é possível atribuir um valor errado



The screenshot shows a code editor window titled "main.ts – typescript". The code defines an interface "Person" with properties "id", "name", and "age?". An array "persons" is initialized with three objects, each having an "id" property that is highlighted in red.

```
1  interface Person {  
2      id: number | string,  
3      name: string,  
4      age?: number  
5  }  
6  
7  const persons: Person[] = [];  
8  
9  persons[0] = { id: 1, name: "Robert Martin", age: 62 };  
10 persons[1] = { id: "2", name: "Martin Fowler", age: 58 };  
11 persons[2] = { id: 3, name: "Vaughn Vernon" };  
12
```

Below the editor, the "PROBLEMS" tab is selected, showing the message "No problems have been detected in the workspace."

As Interfaces tem um resultado
similar aos Types

The screenshot shows a code editor window titled "main.ts – typescript". The code defines an "Entity" interface with an "id" property, which can be either a number or a string. It then defines a "Person" interface that extends "Entity", adding "name" and "age" properties. Finally, it creates a constant "personEntity" of type "Person" with the values { id: 1, name: "Robert Martin", age: 62}. The code editor has tabs for "PROBLEMS", "OUTPUT", and "...", and a search bar at the bottom.

```
TS main.ts X
main.ts — typescript
08

1 interface Entity {
2     id: number | string
3 };
4
5 interface Person extends Entity {
6     name: string,
7     age: number
8 };
9
10 const personEntity: Person = { id: 1, name: "Robert Martin", age: 62};
```

PROBLEMS OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) X

Uma das diferenças é que é possível extender as Interfaces

The screenshot shows a Microsoft Visual Studio Code interface. The main editor window is titled "main.ts — typescript". It contains the following TypeScript code:

```
TS main.ts 1 ●
1 interface Person {
2     id: number | string
3 };
4
5 interface Person {
6     name: string,
7     age: number
8 };
9
10 const person: Person = { name: "Robert Martin", age: 62 };
```

A yellow lightbulb icon is positioned next to the line "const person: Person = {". The word "person" is underlined with a red squiggle. The code editor has a light gray background with syntax highlighting.

Below the editor is a "PROBLEMS" tab with a count of 1, followed by "OUTPUT" and other navigation icons. A search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)" is present. The "PROBLEMS" panel displays one error message:

TS main.ts src 1 > ⚠ Property 'id' is missing in type '{ name: string; age: number; }' but required in type 'Person'. ts(2741) [10, 7]

Também é possível fazer um Merge das interfaces

A screenshot of a TypeScript code editor window titled "main.ts – typescript". The code defines two interfaces, Entity and Person, and a type PersonEntity which is their intersection. A variable personEntity is then declared as an instance of PersonEntity.

```
TS main.ts X
1 interface Entity {
2     id: number | string
3 }
4
5 interface Person {
6     name: string,
7     age: number
8 }
9
10 type PersonEntity = Entity & Person;
11
12 const personEntity: PersonEntity = { id: 1, name: "Robert Martin", age: 62};
```

The "PROBLEMS" tab is selected, showing that there are no problems detected in the workspace.

Intersection Types with Interface



Qual é a diferença entre Type Alias e Interface?

Type aliases and interfaces are **very similar**, and in many cases you can choose between them freely. Almost all features of an interface are available in type, the key distinction is that a type cannot be re-opened to add new properties vs an interface which is always extendable

A screenshot of a Mac OS X application window titled "main.ts – typescript". The window contains a code editor with the file "main.ts" open. The code is as follows:

```
TS main.ts ×  
main.ts  
1 interface Car {  
2     model: string;  
3 }  
4  
5 interface Car {  
6     brand: string;  
7 }  
8  
9 const car: Car = { model: "Toyota", brand: "Corolla" };  
10
```

main.ts — typescript

```
TS main.ts 3 ×
```

```
1 type Car = {  
2     model: string;  
3 }  
4 ⚡  
5 type Car = {  
6     brand: string;  
7 }  
8  
9 const car: Car = { model: "Toyota", brand: "Corolla" };  
10
```

PROBLEMS 3 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) ⚡ ⌂ ⌄ ^ ×

- TS main.ts src 3
 - ✖ Duplicate identifier 'Car'. ts(2300) [Ln 1, Col 6]
 - ✖ Duplicate identifier 'Car'. ts(2300) [Ln 5, Col 6]
 - ✖ Type '{ model: string; brand: string; }' is not assignable to type 'Car'. ts(2322) [Ln 9, Col 37] ⌄
Object literal may only specify known properties, and 'brand' does not exist in type 'Car'.

Item.ts — typescript

TS main.ts TS Item.ts X

```
1  export default abstract class Item {  
2      description: string;  
3      price: number;  
4  
5      constructor (description: string, price: number) {  
6          this.description = description;  
7          this.price = price;  
8      }  
9  
10     abstract calculateTax (): number;  
11 }  
12
```

PROBLEMS OUTPUT TERMINAL ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

No problems have been detected in the workspace.

É possível definir também **classes abstratas**

The screenshot shows a code editor window titled "main.ts — typescript". It contains two tabs: "main.ts" and "Item.ts". The "main.ts" tab is active and displays the following code:

```
1 import Item from "./Item";
2
3 const item = new Item("Guitar", 1000);
4
```

A yellow lightbulb icon is positioned next to the import statement. The line "const item = new Item("Guitar", 1000);" is highlighted with a yellow background, indicating it is the source of the error. In the bottom left corner, the "PROBLEMS" tab is selected, showing one error: "Cannot create an instance of an abstract class. ts(2511) [Ln 3, Col 14]".

Não é possível instanciar classes
abstratas

```
1 import Item from "./Item";
2
3 export default class Guitar extends Item {
4
5     constructor(price: number) {
6         super("Guitar", price);
7     }
8
9     calculateTax(): number {
10        return (this.price * 10)/100;
11    }
12}
```

PROBLEMS OUTPUT TERMINAL ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

No problems have been detected in the workspace.

Ao extender uma classe abstrata **devemos implementar os métodos abstratos**

```
main.ts — typescript
TS main.ts X TS Item.ts TS Guitar.ts
1 import Guitar from "./Guitar";
2
3 const guitar = new Guitar(1000);
4
```

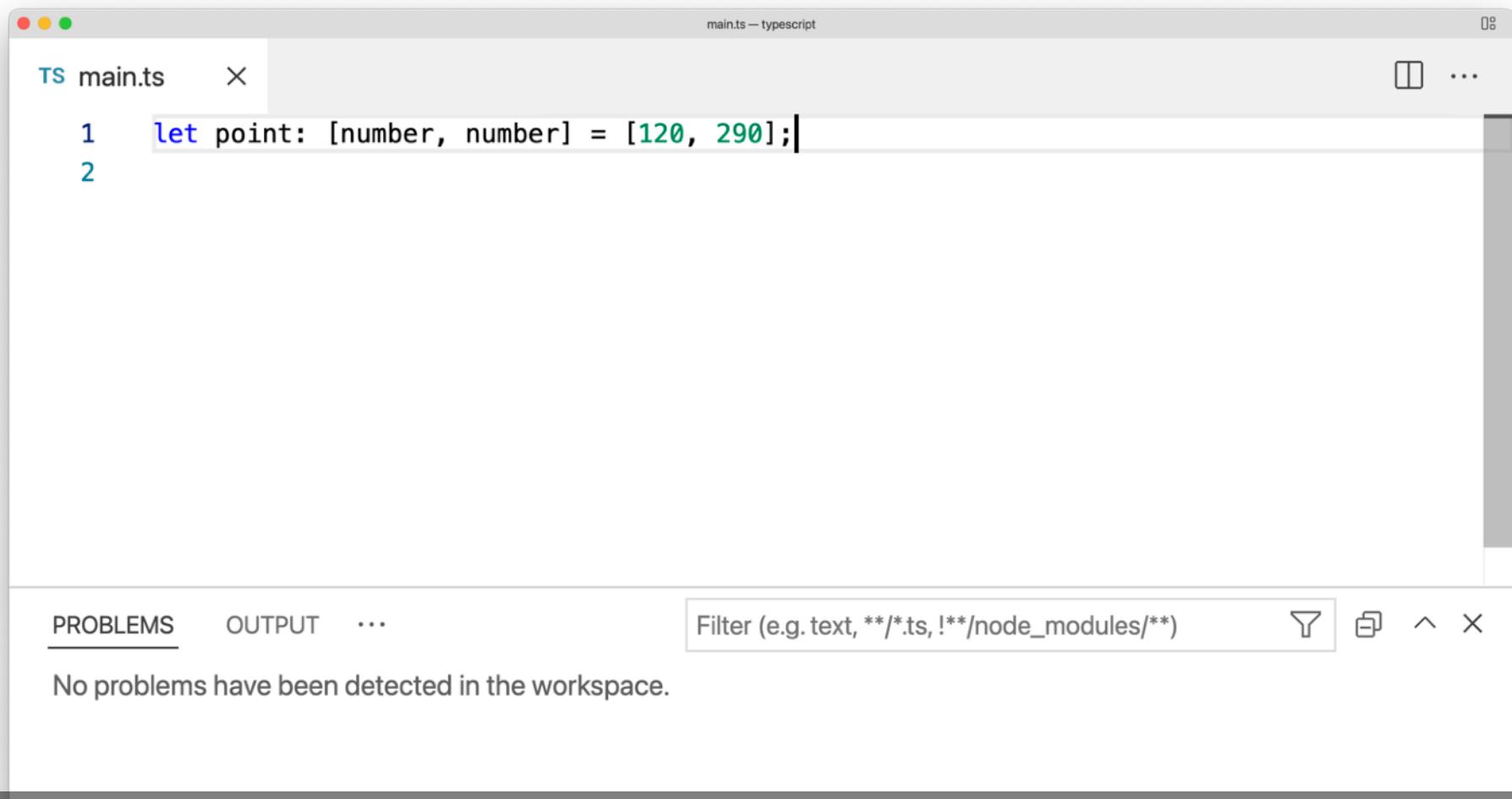
PROBLEMS OUTPUT TERMINAL ... Filter (e.g. text, **/*.ts, !**/node_modules/**) ✖

No problems have been detected in the workspace.

A classe que **extende** a classe
abstrata pode ser instanciada



Qual é a diferença entre **abstract class**
e **interface**?



A screenshot of a code editor window titled "main.ts — typescript". The code editor displays the following TypeScript code:

```
1 let point: [number, number] = [120, 290];
```

The code editor interface includes a toolbar at the top with icons for file operations, and a bottom navigation bar with tabs for "PROBLEMS", "OUTPUT", and "...". The "PROBLEMS" tab is active, showing the message: "No problems have been detected in the workspace."

É possível também definir Tuplas que
são arrays com elementos tipados

```
main.ts — typescript
```

```
TS main.ts ×
```

```
1 let point: [number, number];
2 point = [100, 200];
```

```
PROBLEMS OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) ⚏ ☰ ⌂ ⌄ ⌁ ✕
```

No problems have been detected in the workspace.

Um bom exemplo são pontos cartesianos ou latitude e longitude

The screenshot shows a code editor window titled "main.ts – typescript". The code in the editor is:

```
1 let point: [number, number];
2 point = [100, "a"];
```

The line "point = [100, "a"];" is highlighted with a yellow background. The word "a" is underlined with a red squiggly line, indicating a type error. In the bottom left, the "PROBLEMS" tab is selected, showing 1 error. The error message in the Problems panel is:

TS main.ts src 1

⊗ Type 'string' is not assignable to type 'number'. ts(2322) [Ln 2, Col 15]

Tanto os tipos quanto o número de elementos deve ser considerado

The screenshot shows a code editor window titled "main.ts — typescript". The code in the editor is:

```
1 let point: [number, number];
2 point = [1, 2, 3];
```

The line "point = [1, 2, 3];" has a red wavy underline under the variable name "point".

Below the editor, the "PROBLEMS" tab is selected, showing 1 problem. The problem details are:

TS main.ts src 1

⊗ Type '[number, number, number]' is not assignable to type '[number, nu... ts(2322) [Ln 2, Col 1] ^
Source has 3 element(s) but target allows only 2.

Tanto os tipos quanto o número de elementos deve ser considerado

The screenshot shows a code editor window titled "main.ts – typescript". The code in the editor is:

```
1 let point: [number, number];
2 point = [100];
```

The line "point = [100];" is highlighted with a yellow background, and the word "point" is underlined with a red wavy line, indicating a TypeScript error. Below the editor, the "PROBLEMS" tab is selected, showing one error for "main.ts src":

TS main.ts src 1

×

Type '[number]' is not assignable to type '[number, number]'. ts(2322) [Ln 2, Col 1] ^
Source has 1 element(s) but target requires 2.

Tanto os tipos quanto o número de elementos deve ser considerado

Modificadores de Visibilidade

Existem diversos modificadores no TypeScript como public, private e protected, sendo public é o padrão

The screenshot shows a code editor window titled "main.ts – typescript". The file content is as follows:

```
1  class Person {
2      public name: string;
3      public age: number;
4
5      constructor (name: string, age: number) {
6          this.name = name;
7          this.age = age;
8      }
9  }
10
11 const person = new Person("Robert Martin", 62);
12 console.log(person.name, person.age);
```

The code defines a `Person` class with `name` and `age` properties. It also includes a constructor that initializes these properties. Finally, it creates a `person` object and logs its `name` and `age` to the console.

Below the code editor, there is a "PROBLEMS" tab, an "OUTPUT" tab, a "TERMINAL" tab, and a "..." tab. A "Filter (e.g. text, **/*.ts, !**/node_modules/**)" input field is present, along with a search icon and other UI elements. The status bar at the bottom displays the message "No problems have been detected in the workspace."

O modificador `public` permite que a propriedade seja acessada externamente

The screenshot shows a code editor window titled "main.ts – typescript". The code in the main.ts file is:

```
1  class Person {
2      name: string;
3      age: number;
4
5      constructor (name: string, age: number) {
6          this.name = name;
7          this.age = age;
8      }
9  }
10
11 const person = new Person("Robert Martin", 62);
12 console.log(person.name, person.age);
```

Below the code editor, the "PROBLEMS" tab is selected, showing the message "No problems have been detected in the workspace."

Se nada for definido o modificador
utilizado é public

The screenshot shows a code editor window titled "main.ts — typescript". The code defines a class "Person" with a private property "age". The line "console.log(person.name, person.age);" is highlighted in yellow, indicating a potential issue. The "PROBLEMS" tab is active, showing one error: "Property 'age' is private and only accessible within class 'Person'. ts(2341) [Ln 12, Col 33]".

```
1  class Person {  
2      public name: string;  
3      private age: number;  
4  
5      constructor (name: string, age: number) {  
6          this.name = name;  
7          this.age = age;  
8      }  
9  }  
10  
11 const person = new Person("Robert Martin", 62);  
12 console.log(person.name, person.age);
```

PROBLEMS 1 OUTPUT TERMINAL ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

TS main.ts src 1

⊗ Property 'age' is private and only accessible within class 'Person'. ts(2341) [Ln 12, Col 33]

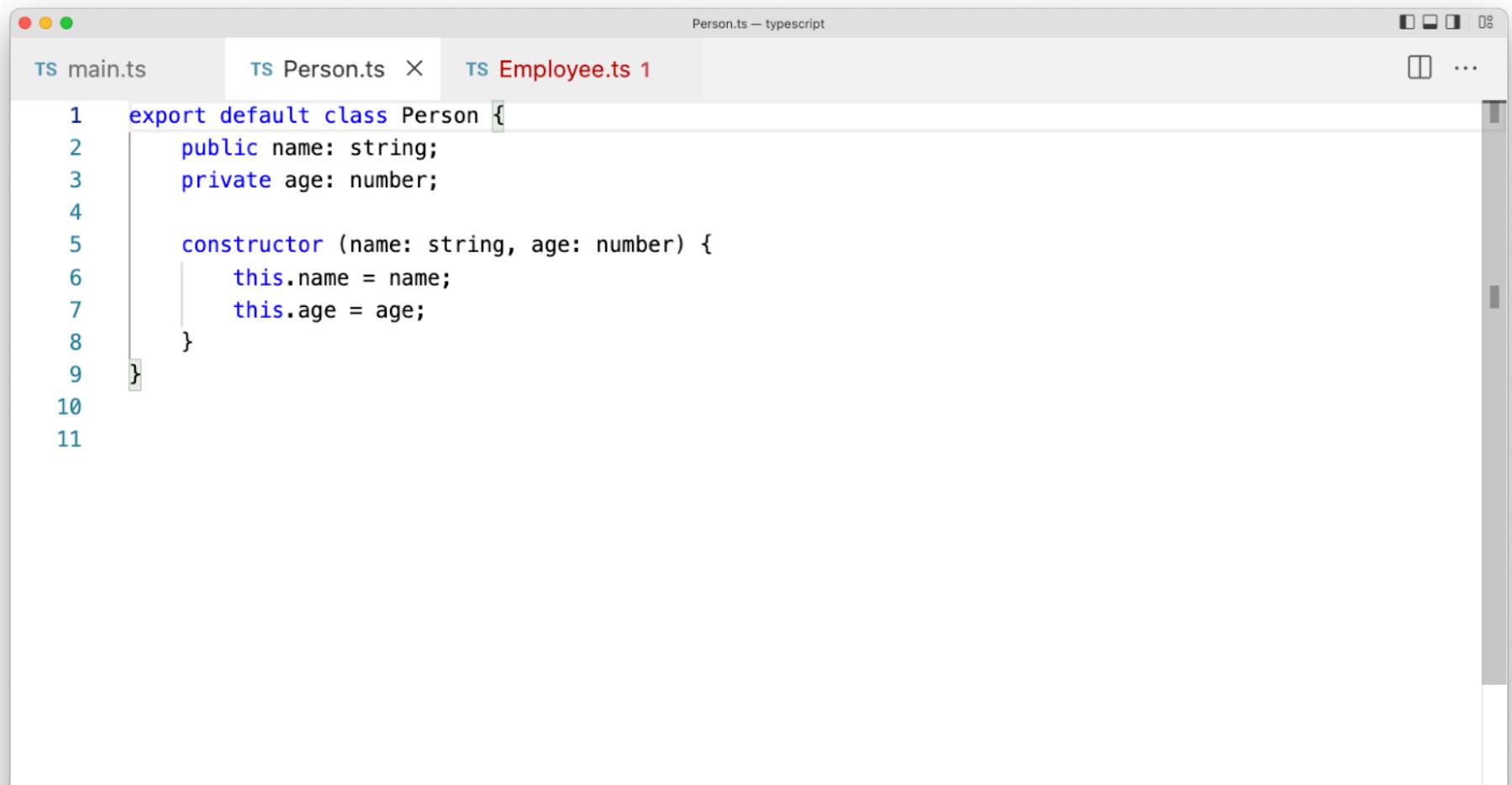
O modificador `private` torna as propriedades inacessíveis

The screenshot shows a code editor window titled "main.ts – typescript". The code is as follows:

```
1  class Person {
2      public name: string;
3      #age: number;
4
5      constructor (name: string, age: number) {
6          this.name = name;
7          this.#age = age;
8      }
9  }
10
11 const person = new Person("Robert Martin", 62);
12 console.log(person.name, person.#age);
```

The line "console.log(person.name, person.#age);" is highlighted in yellow. In the bottom left, the "PROBLEMS" tab has a count of 1, and the list shows a single error: "Property '#age' is not accessible outside class 'Person' because it has a private identifier. ts(18013) [Ln 12, Col 33]".

Também é possível utilizar o # como modificador privado



The screenshot shows a code editor window titled "Person.ts – typescript". The tabs at the top are "main.ts", "Person.ts", and "Employee.ts 1". The "Person.ts" tab is active. The code in "Person.ts" is:

```
1  export default class Person {
2      public name: string;
3      private age: number;
4
5      constructor (name: string, age: number) {
6          this.name = name;
7          this.age = age;
8      }
9
10 }
11
```

Além disso, o modificador `private` também
restringe as propriedades em subclasses

The screenshot shows a code editor window with three tabs: main.ts, Person.ts, and Employee.ts. The Employee.ts tab is active, displaying the following code:

```
1 import Person from "./Person";
2
3 export default class Employee extends Person {
4     salary: number;
5
6     constructor (name: string, age: number, salary: number) {
7         super(name, age);
8         this.salary = salary;
9         console.log(this.age);
10    }
11 }
12
```

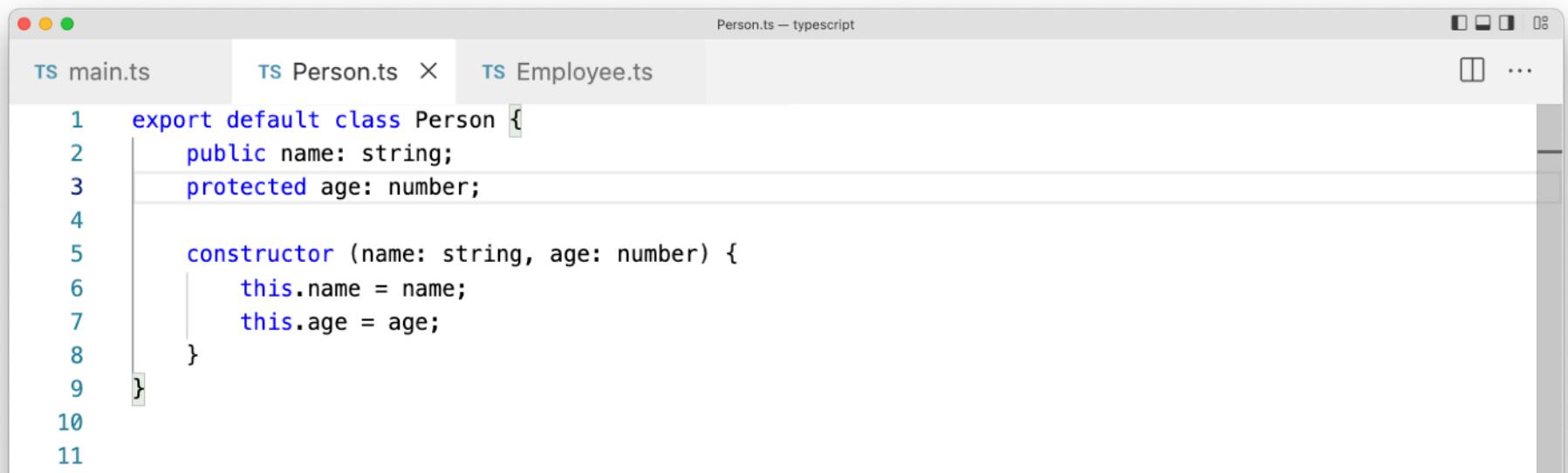
A yellow highlight covers the entire code block. In the bottom right corner of the code area, there is a small red square icon.

Below the code editor is a navigation bar with PROBLEMS (1), OUTPUT, TERMINAL, and other icons. The PROBLEMS tab is selected, showing one error:

TS Employee.ts src 1

Property 'age' is private and only accessible within class 'Person'. ts(2341) [Ln 9, Col 20]

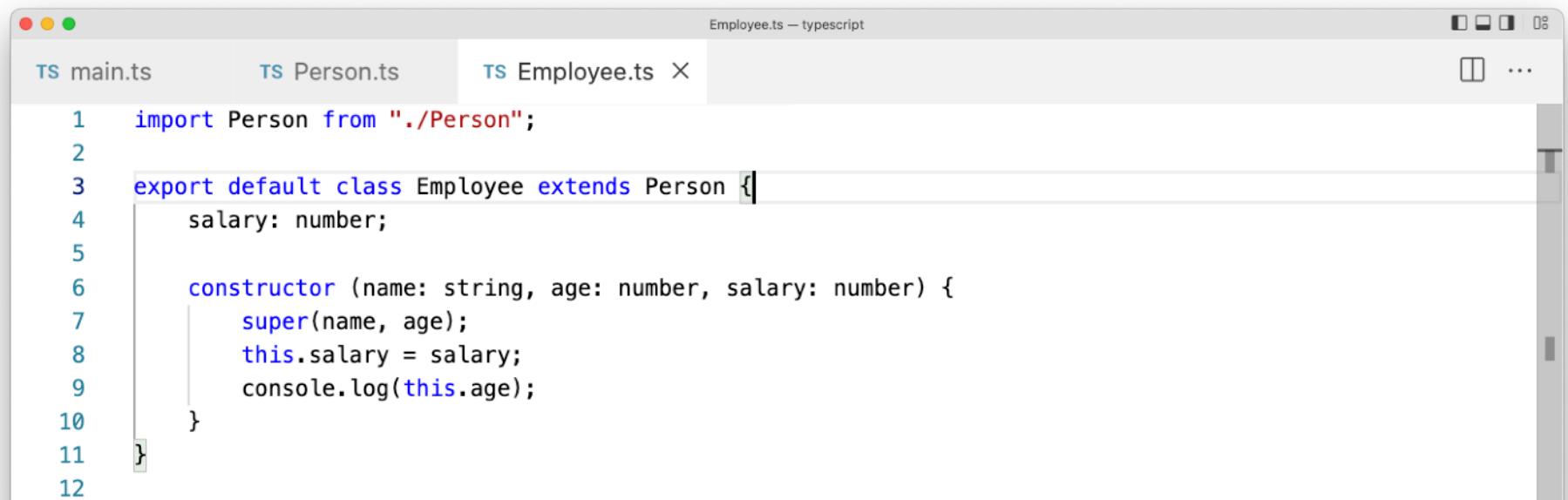
Não é possível acessar e nem
modificar



The screenshot shows a Mac OS X window titled "Person.ts — typescript". It contains three tabs: "main.ts", "Person.ts" (which is selected and highlighted in blue), and "Employee.ts". The content of the "Person.ts" tab is as follows:

```
1  export default class Person {
2      public name: string;
3      protected age: number;
4
5      constructor (name: string, age: number) {
6          this.name = name;
7          this.age = age;
8      }
9
10
11
```

Existem também o modificador
protected



```
Employee.ts — typescript
TS main.ts      TS Person.ts      TS Employee.ts ×
1 import Person from "./Person";
2
3 export default class Employee extends Person {
4     salary: number;
5
6     constructor (name: string, age: number, salary: number) {
7         super(name, age);
8         this.salary = salary;
9         console.log(this.age);
10    }
11 }
12
```

Com ele é possível acessar as propriedades nas subclasses

The screenshot shows a code editor window titled "main.ts – typescript". The main pane displays the following TypeScript code:

```
1 import Person from "./Person";
2
3 const person = new Person("Robert Martin", 62);
4 console.log(person.name, person.age);
5
```

The word "age" is underlined with a red squiggly line, indicating a TypeScript error. The code editor's status bar at the bottom shows "PROBLEMS 1", "OUTPUT", "TERMINAL", and "...". The "PROBLEMS" tab is active, showing one error: "Property 'age' is protected and only accessible within class 'Person' and its subclasses. ts(2445) [Ln 4, Col 33]".

Mas não é possível acessar de fora,
por meio da instância

Readonly

O modificador `readonly` impede que uma propriedade seja modificada após ser atribuída

A screenshot of a code editor window titled "Invoice.ts — typescript". The code editor displays the following TypeScript code:

```
1  export default class Invoice {
2
3      constructor (readonly amount: number) {
4
5  }
6
7  const invoice = new Invoice(1000);
8  invoice.amount = 100;
9
10
```

The line "invoice.amount = 100;" is highlighted with a yellow background, indicating it is the source of the error. The word "amount" is underlined with a red wavy line, signifying it is a read-only property.

Below the code editor, the "PROBLEMS" tab is selected, showing one error: "Cannot assign to 'amount' because it is a read-only property. ts(2540) [Ln 8, Col 9]".

readonly

any e unknown

Muitas vezes não sabemos exatamente o tipo que queremos utilizar, nesse caso existem algumas possibilidades como o any e o unknown

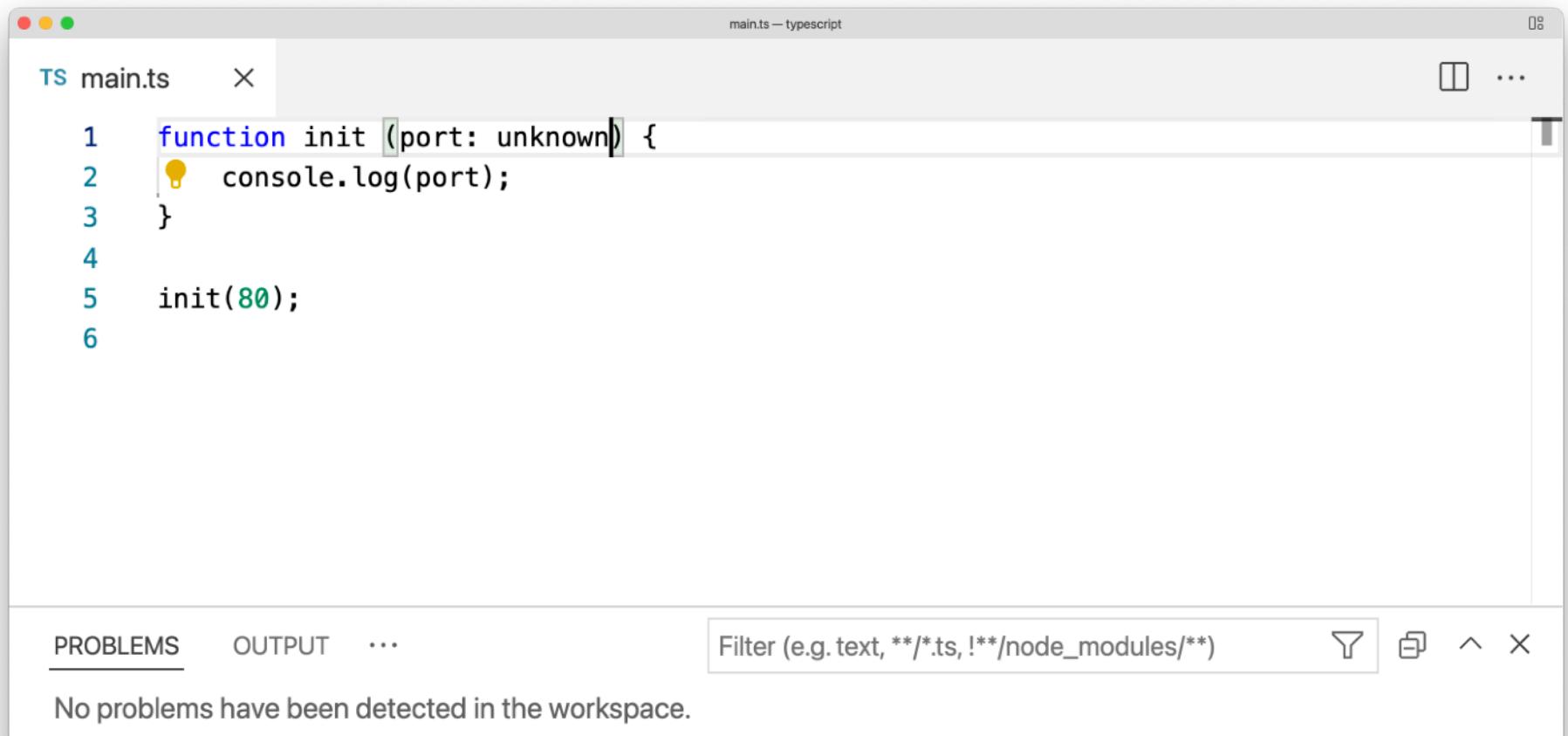


The screenshot shows a Microsoft Visual Studio Code interface. The title bar says "main.ts — typescript". The editor tab bar has "main.ts" with a "TS" icon. The code in the editor is:

```
1  function init (port: any) {  
2      console.log(port);  
3  }  
4  
5  init(80);  
6
```

The number 80 is highlighted in green. Below the editor is a "PROBLEMS" tab, which is underlined, indicating it is active. The output area below shows the message: "No problems have been detected in the workspace."

O any indica que o tipo pode ser
qualquer coisa



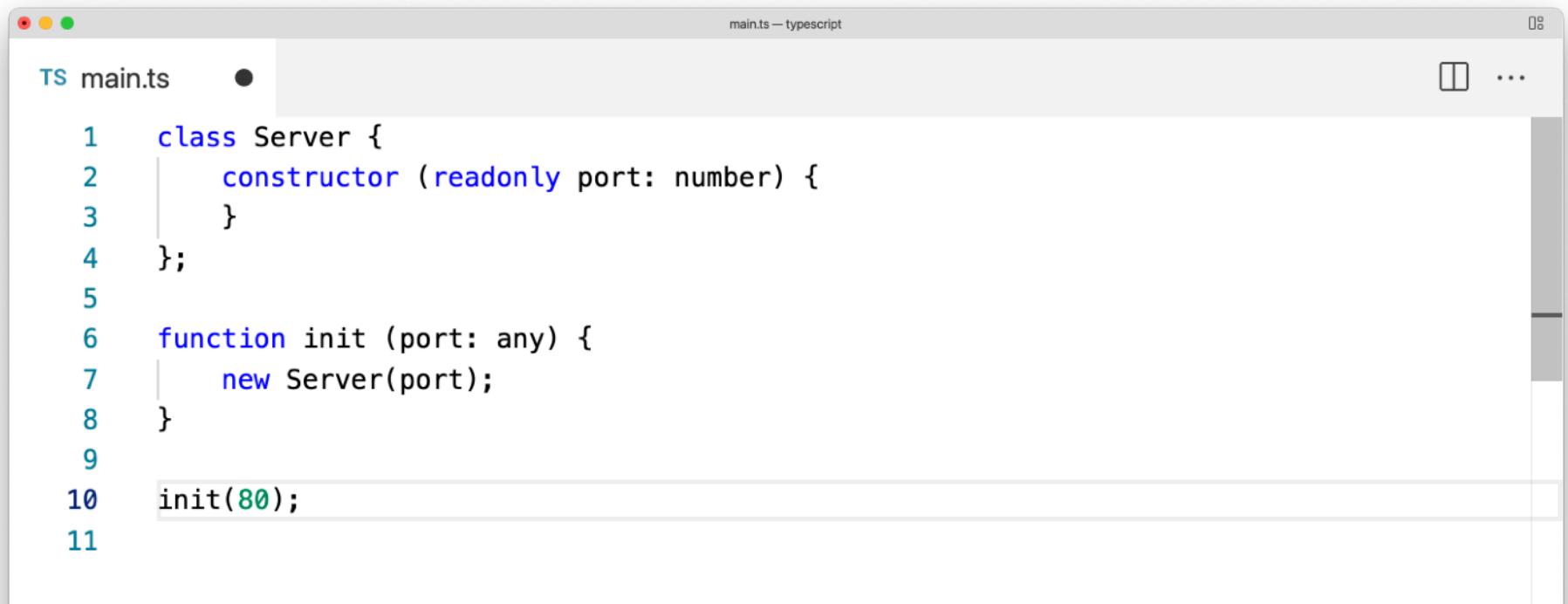
The screenshot shows a code editor window titled "main.ts – typescript". The code in the editor is:

```
1  function init (port: unknown) {
2    console.log(port);
3  }
4
5  init(80);
6
```

A yellow lightbulb icon is shown next to the line "console.log(port);", indicating a potential issue or suggestion related to the use of the `unknown` type.

Below the editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, and ..., followed by a search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)". The "PROBLEMS" tab is currently selected. The status bar at the bottom of the editor window displays the text "No problems have been detected in the workspace."

O `unknown` indica que o tipo é
desconhecido



```
main.ts  main.ts — typescript
```

```
1  class Server {
2    constructor (readonly port: number) {
3    }
4  };
5
6  function init (port: any) {
7    new Server(port);
8  }
9
10 init(80);
11
```

PROBLEMS OUTPUT ...

Filter (e.g. text, **/*.ts, !**/node_modules/**)



No problems have been detected in the workspace.

O `any` simplesmente desliga o type-checking

The screenshot shows a code editor window for a file named `main.ts` with the extension `-typescript`. The code contains a class `Server` and a function `init`. The variable `port` is declared as `unknown` in the `init` function, but it is used as a parameter for the `Server` constructor, which expects a `number`. This results in a TypeScript error highlighted by a red squiggle under `port` in the `new Server(port);` line.

```
TS main.ts 1
1  class Server {
2    constructor (readonly port: number) {
3    }
4  };
5
6  function init (port: unknown) {
7    new Server(port);
8  }
9
10 init(80);
11
```

Below the code editor, the `PROBLEMS` tab is active, showing one error. The error message is: `Argument of type 'unknown' is not assignable to parameter of type 'number'. ts(2345) [7, 13]`.

O `unknown` não permite que a variável seja utilizada em outro tipo

never

O tipo never indica que algo nunca vai acontecer, por exemplo, o retorno de uma function

The screenshot shows a macOS terminal window with a light gray background. At the top, there's a title bar with the text "main.ts — typescript". Below the title bar is a tab bar with three tabs: "main.ts" (which is active), "X", and "...". The main content area contains the following TypeScript code:

```
1 function init (): never {  
2     throw new Error("no return");  
3 }  
4  
5 init();  
6
```

Below the code editor, there's a navigation bar with several tabs: "PROBLEMS", "OUTPUT", "TERMINAL" (which is underlined, indicating it's the active tab), and "DEBUG CONSOLE". To the right of the tabs are several icons: a square with a right-pointing arrow, "zsh", a plus sign, a downward arrow, a square with a diagonal line, a trash can, an upward arrow, and an 'X'.

In the bottom left corner of the terminal window, there's a prompt: "rodrigobranas typescript \$".

O never nunca retorna

A screenshot of a terminal window titled "main.ts — typescript". The code editor shows a file named "main.ts" with the following content:

```
1 function loop (): never {
2     let i = 0;
3     while(true) {
4         console.log(i++);
5     }
6 }
7
8 loop();
9
```

The terminal below shows the command "rodrigobranas typescript \$" followed by a cursor icon.

O never nunca retorna



Qual é a diferença entre **void** e **never**?

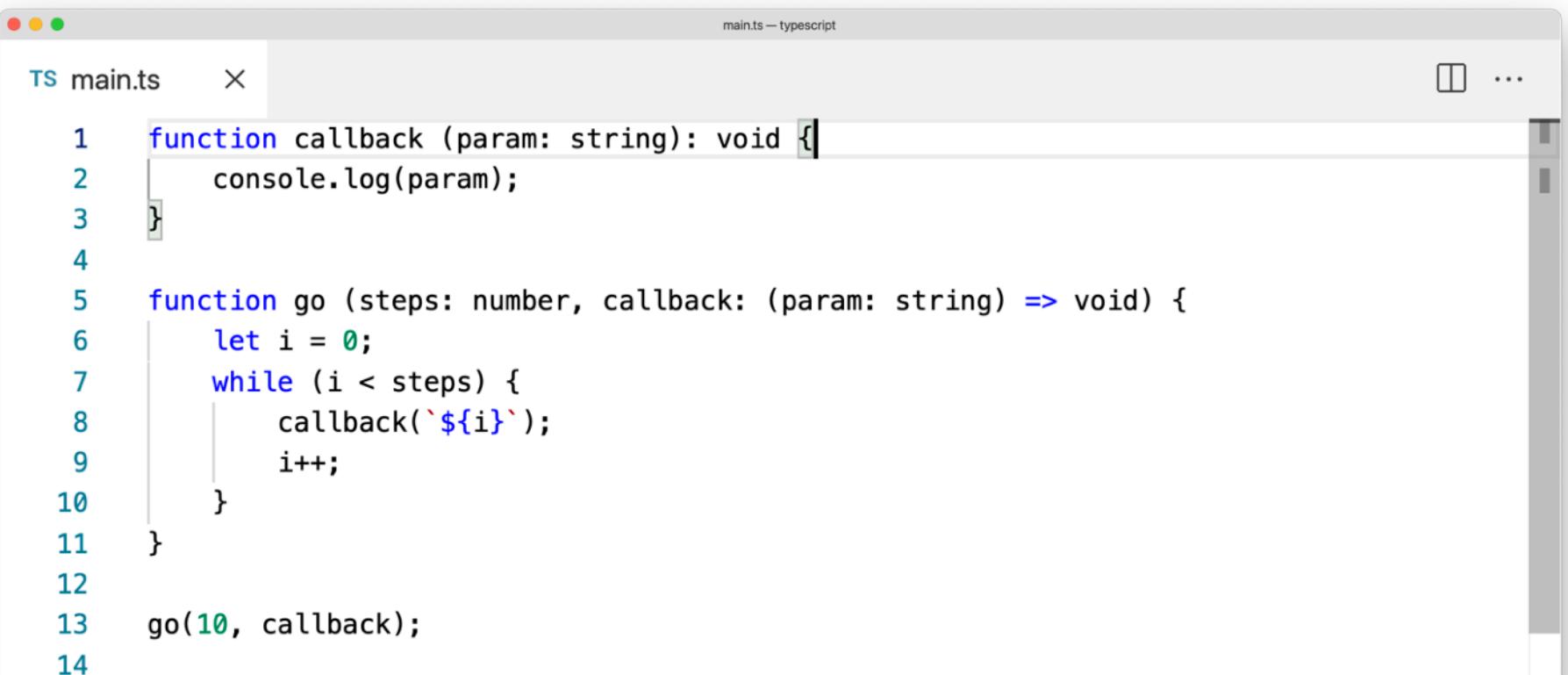
Function

Parâmetros podem ser do tipo Function

The screenshot shows a Mac OS X window titled "main.ts — typescript". The window contains a code editor with the following TypeScript code:

```
1  function callback (param: string): void {
2      console.log(param);
3  }
4
5  function go (steps: number, callback: Function) {
6      let i = 0;
7      while (i < steps) {
8          callback(` ${i}`);
9          i++;
10     }
11 }
12
13 go(10, callback);
14
```

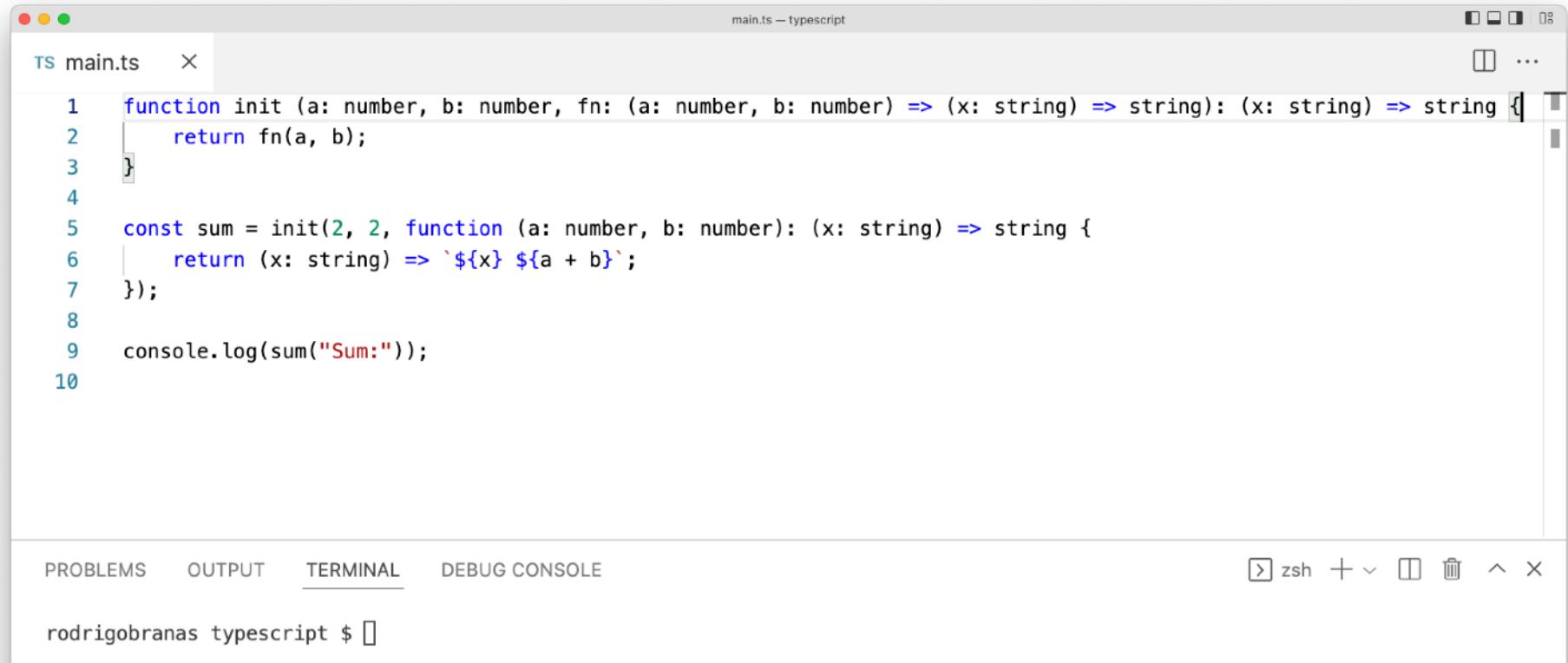
Function



```
main.ts  ×  main.ts — typescript
```

```
1  function callback (param: string): void {
2      console.log(param);
3  }
4
5  function go (steps: number, callback: (param: string) => void) {
6      let i = 0;
7      while (i < steps) {
8          callback(` ${i}`);
9          i++;
10     }
11 }
12
13 go(10, callback);
14
```

Definição de parâmetro como uma
definição de função



A screenshot of a code editor window titled "main.ts — typescript". The code in the editor is:

```
TS main.ts ×
1 function init (a: number, b: number, fn: (a: number, b: number) => (x: string) => string): (x: string) => string {
2     return fn(a, b);
3 }
4
5 const sum = init(2, 2, function (a: number, b: number): (x: string) => string {
6     return (x: string) => `${x} ${a + b}`;
7 });
8
9 console.log(sum("Sum:"));
10
```

The editor has tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is active, showing the command "rodrigobranas typescript \$".

Definição de parâmetro como uma definição de função

Suppress Comments

Muitas vezes é necessário desligar a verificação do TypeScript em determinados pontos do código-fonte e para isso é possível utilizar comentários que orientam o transpiler

main.ts — typescript

```
TS main.ts 1 ×
```

```
1  function init (port) {
2    console.log(port);
3  }
4
5  init(80);
6
7
8
9
10
11
12
```

PROBLEMS 1 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

TS main.ts src 1

Parameter 'port' implicitly has an 'any' type. ts(7006) [1,16]

The screenshot shows a code editor window with a TypeScript file named `main.ts`. The file contains the following code:

```
// @ts-ignore
function init (port) {
    console.log(port);
}

init(80);
```

The code editor interface includes a tab bar at the top with `main.ts`, a close button, and a menu icon. Below the editor is a status bar with tabs for `PROBLEMS`, `OUTPUT`, and `...`. A search bar labeled `Filter (e.g. text, **/*.ts, !**/node_modules/**)` is also present. The bottom of the window displays the message `No problems have been detected in the workspace.`

main.ts — typescript

```
TS main.ts 1 ●
```

```
1 // @ts-ignore
2 function init (port) {
3   console.log(port);
4   const url: string = true;
5 }
6
7 init(80);
8
9
10
11
12
```

PROBLEMS 1 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

TS main.ts src 1

✖ Type 'boolean' is not assignable to type 'string'. ts(2322) [4, 8]

main.ts — typescript

TS main.ts X

```
1 // @ts-ignore
2 function init (port) {
3     console.log(port);
4     // @ts-ignore
5     const url: string = true;
6 }
7
8 init(80);
9
10
11
12
```

PROBLEMS OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

No problems have been detected in the workspace.

The screenshot shows a Mac OS X application window with a light gray background. At the top, there's a title bar with the text "main.ts — typescript". On the left side of the main area, there's a tab labeled "TS main.ts" with a small circular icon next to it. The main content area contains the following TypeScript code:

```
1 // @ts-nocheck
2 function init (port) {
3     console.log(port);
4     const url: string = true;
5 }
6
7 init(80);
8
9
10
11
12
```

The code uses standard TypeScript syntax, including a type annotation for the `url` variable. There are no errors or warnings shown in the code editor.

At the bottom of the window, there's a toolbar with three tabs: "PROBLEMS" (underlined), "OUTPUT", and "...". To the right of the tabs is a search bar with the placeholder text "Filter (e.g. text, **/*.ts, !**/node_modules/**)". Further to the right are icons for filtering, expanding/collapsing, and closing the panel.

The status bar at the bottom of the window displays the text "No problems have been detected in the workspace.".

Generics

É possível tipar classes permitindo um controle ainda maior sobre a interação com os objetos, inclusive com herança

The screenshot shows a code editor window titled "main.ts — typescript". The code is a simple TypeScript class definition:

```
1  class Repository {
2      list: any;
3
4      constructor () {
5          this.list = [];
6      }
7
8      add (element: any) {
9          this.list.push(element);
10 }
11 }
12
```

The code editor interface includes tabs for "PROBLEMS", "OUTPUT", and "...", and a search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)". Below the editor, a message states "No problems have been detected in the workspace."

The screenshot shows a code editor window titled "main.ts — typescript". The file contains the following TypeScript code:

```
1  class Repository<T> {
2      list: T[];
3
4      constructor () {
5          this.list = [];
6      }
7
8      add (element: T) {
9          this.list.push(element);
10     }
11 }
12
```

The code defines a generic class `Repository` that stores a list of type `T`. It has a constructor that initializes the list to an empty array, and an `add` method that pushes an element onto the list.

Below the code editor, there is a navigation bar with tabs for "PROBLEMS", "OUTPUT", and "...". The "PROBLEMS" tab is currently selected. To its right is a search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)" with a magnifying glass icon. Further to the right are icons for expanding/collapsing the sidebar, closing the panel, and other controls.

The message "No problems have been detected in the workspace." is displayed below the navigation bar.

main.ts — typescript

TS main.ts 1 ●

```
12
13  class Person {
14      constructor (readonly name: string, readonly age: number) {
15      }
16  }
17
18  const persons = new Repository<Person>();
19  persons.add(new Person("Robert Martin", 62));
20  persons.add(new Person("Martin Fowler", 58));
21  persons.add(JavaScript);
22
```

PROBLEMS 1 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

TS main.ts src 1

✖ Argument of type 'string' is not assignable to parameter of type 'Person'. ts(2345) [21, 13]

The screenshot shows a code editor window titled "main.ts — typescript". The file contains the following TypeScript code:

```
1 class Repository<T extends Person> {
2     list: T[];
3
4     constructor () {
5         this.list = [];
6     }
7
8     add (element: T) {
9         this.list.push(element);
10    }
11 }
12
```

The code defines a generic class `Repository` that extends `Person`. It has a private field `list` of type `T[]` and a constructor that initializes it to an empty array. It also has a public method `add` that takes an element of type `T` and adds it to the list.

Below the code editor, there is a "PROBLEMS" tab, an "OUTPUT" tab, and a "..." tab. A search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)" is present, along with icons for filter, refresh, collapse, and close.

No problems have been detected in the workspace.

The screenshot shows a code editor window titled "main.ts — typescript". The code is written in TypeScript and defines a class "Author" extending "Person". It includes a constructor taking name, age, and a readonly array of books (titles). Two authors are added to a Repository: Robert Martin (62, Clean Code, Clean Architecture) and Martin Fowler (58, Refactoring).

```
18  class Author extends Person {
19    constructor(name: string, age: number, readonly books: { title: string }[]) {
20      super(name, age);
21    }
22  }
23
24 const authors = new Repository<Author>();
25 authors.add(new Author("Robert Martin", 62, [ { title: "Clean Code" }, { title: "Clean Architecture" } ]));
26 authors.add(new Author("Martin Fowler", 58, [ { title: "Refactoring" } ]));
27
28
```

Below the editor, the "PROBLEMS" tab is active, showing the message: "No problems have been detected in the workspace."

TS main.ts 1 ●

```
17
18     class Author extends Person {
19         constructor (name: string, age: number, readonly books: { title: string }[]) {
20             super(name, age);
21         }
22     }
23
24     const authors = new Repository<Author>();
25     authors.add(new Author("Robert Martin", 62, [ { title: "Clean Code" }, { title: "Clean Code Refactored" } ]));
26     authors.add(new Author("Martin Fowler", 58, [ { title: "Refactoring" } ]));
27     const booleans = new Repository<boolean>();
28 
```

PROBLEMS 1 OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**) ✖ ^ ×

TS main.ts src 1

✖ Type 'boolean' does not satisfy the constraint 'Person'. ts(2344) [27, 33]

Modules

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

main.ts — typescript

TS main.ts 4 × TS Repository.ts 1 TS Author.ts 1 TS Person.ts

```
1 const authors = new Repository<Author>();  
2 authors.add(new Author("Robert Martin", 62, [ { title: "Clean Code" }, { title: "Clean  
3 authors.add(new Author("Martin Fowler", 58, [ { title: "Refactoring" }]));  
4
```

PROBLEMS 6 OUTPUT ...

Filter (e.g. text, **/*.ts, !**/node_modules/**)

- > TS Author.ts src 1
- ✗ TS main.ts src 4
 - 💡 Cannot find name 'Repository'. ts(2304) [1,21]
 - ✗ Cannot find name 'Author'. ts(2304) [1,32]
 - ✗ Cannot find name 'Author'. Did you mean 'Author'? ts(2552) [2,17]

Separando as classes em arquivos diferentes

The screenshot shows the VS Code interface with the following details:

- File Tabs:** Repository.ts — typescript (active), main.ts (4 errors), Author.ts (1 error), Person.ts.
- Code Content (Repository.ts):**

```
1  export class Repository<T extends Person> {
2      list: T[];
3
4      constructor () {
5          this.list = [];
6      }
7
8      add (element: T) {
9          this.list.push(element);
10     }
11 }
12
```

- Bottom Bar:** PROBLEMS (6), OUTPUT, ...
- Problems View:** Filter (e.g. text, **/*.ts, !**/node_modules/**)

 - > TS Author.ts src (1)
 - > TS main.ts src (4)
 - < TS Repository.ts src (1)

- Status Bar:** Cannot find name 'Person'. ts(2304) [1, 35]

Separando as classes em arquivos diferentes e exportando

The screenshot shows a code editor window with four tabs: main.ts (4 errors), Repository.ts (1 error), Author.ts (1 error, currently active), and Person.ts. The Author.ts tab contains the following code:

```
1 export class Author extends Person {
2     constructor (name: string, age: number, readonly books: { title: string }[]) {
3         super(name, age);
4     }
5 }
```

The Person class is highlighted with a yellow background. In the bottom left, the PROBLEMS tab shows 6 errors. The first error is for Author.ts: "Cannot find name 'Person'. ts(2304) [1, 29]". Other errors listed are for main.ts (4) and Repository.ts (1).

Separando as classes em arquivos diferentes e exportando

The screenshot shows a VS Code interface with the following details:

- Editor Tab Bar:** Contains tabs for `main.ts`, `Repository.ts`, `Author.ts`, and `Person.ts`. `Person.ts` is the active tab.
- Editor Content:** The `Person.ts` file contains the following code:

```
1 export class Person {  
2     constructor (readonly name: string, readonly age: number) {  
3     }  
4 }  
5
```
- PROBLEMS Panel:** Shows 6 issues. One issue is highlighted:
 - Author.ts src (1)**: `Cannot find name 'Person'. ts(2304) [1, 29]`
 - main.ts src (4)**: None listed.
 - Repository.ts src (1)**: None listed.
- Bottom Status Bar:** Shows the text "Separando as classes em arquivos diferentes e exportando".

Separando as classes em arquivos
diferentes e exportando

The screenshot shows a code editor window with four tabs at the top: main.ts (4 errors), Repository.ts (1 error), Author.ts (the active tab), and Person.ts. The Author.ts file contains the following TypeScript code:

```
1 import { Person } from "./Person";
2
3 export class Author extends Person {
4     constructor (name: string, age: number, readonly books: { title: string }[]) {
5         super(name, age);
6     }
7 }
```

Below the editor is a navigation bar with PROBLEMS (5), OUTPUT, and a search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)". Under the PROBLEMS tab, there are two entries: main.ts src (4 errors) and Repository.ts src (1 error).

Importando as classes

A screenshot of a code editor window titled "Repository.ts — typescript". The window shows the following TypeScript code:

```
1 import { Person } from "./Person";
2
3 export class Repository<T extends Person> {
4     list: T[];
5
6     constructor () {
7         this.list = [];
8     }
9
10    add (element: T) {
11        this.list.push(element);
12    }

```

The code defines a generic class `Repository` that extends the `Person` class. It has a private list of type `T[]` and a public `add` method that adds elements of type `T` to the list.

Below the code editor, there is a navigation bar with tabs for "PROBLEMS" (4), "OUTPUT", and "...". There is also a search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)" and a toolbar with icons for filter, expand, collapse, and close.

Importando as classes

The screenshot shows a code editor window with four tabs: main.ts, Repository.ts, Author.ts, and Person.ts. The main.ts tab is active, displaying the following TypeScript code:

```
1 import { Repository } from "./Repository";
2 import { Author } from "./Author";
3
4 const authors = new Repository<Author>();
5 authors.add(new Author("Robert Martin", 62, [ { title: "Clean Code" }, { title: "Clean
6 authors.add(new Author("Martin Fowler", 58, [ { title: "Refactoring" }]));
7
```

The code imports the Repository and Author classes from their respective files. It then creates a new Repository instance for the Author class and adds two authors to it.

Below the code editor is a status bar with tabs for PROBLEMS, OUTPUT, and The PROBLEMS tab is active, showing the message: "No problems have been detected in the workspace."

Importando as classes

The screenshot shows a code editor window with four tabs at the top: main.ts (active), Repository.ts, Author.ts, and Person.ts. The main.ts tab contains the following TypeScript code:

```
1 import Repository from "./Repository";
2 import Author from "./Author";
3
4 const authors = new Repository<Author>();
5 authors.add(new Author("Robert Martin", 62, [ { title: "Clean Code" }, { title: "Clean
6 authors.add(new Author("Martin Fowler", 58, [ { title: "Refactoring" }]));
7
```

Below the code editor is a navigation bar with PROBLEMS (2), OUTPUT, and a filter input field labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)". The PROBLEMS section lists two errors:

- Module ''/Users/rodrigobranas/development/workspace/branas/fullstackjs/typescript/src/Rep... ts(2613) [1, 8]
- Module ''/Users/rodrigobranas/development/workspace/branas/fullstackjs/typescript/src/Aut... ts(2613) [2, 8]

Importando as classes por padrão

The screenshot shows a code editor interface with several tabs open. The active tab is `Repository.ts`, which contains the following TypeScript code:

```
1 import { Person } from "./Person";
2
3 export default class Repository<T extends Person> {
4     list: T[];
5
6     constructor () {
7         this.list = [];
8     }
9
10    add (element: T) {
11        this.list.push(element);
12    }
}
```

Below the code editor, there is a navigation bar with tabs for `PROBLEMS`, `OUTPUT`, and `...`. The `PROBLEMS` tab is selected, showing one error: `Module ''/Users/rodrigobranas/development/workspace/branas/fullstackjs/typescript/src/Author.ts' has no exports`. The error message is preceded by a red circle with the number `1`.

Exportando as classes por padrão

The screenshot shows a code editor window with two tabs open: `Author.ts` and `Person.ts`. The `Author.ts` tab is active, displaying the following code:

```
1 import { Person } from "./Person";
2
3 export default class Author extends Person {
4     constructor (name: string, age: number, readonly books: { title: string }[]) {
5         super(name, age);
6     }
7 }
```

The `Person.ts` tab is also visible but contains no code. Below the editor, the `PROBLEMS` tab is selected, showing the message: "No problems have been detected in the workspace."

Exportando as classes por padrão

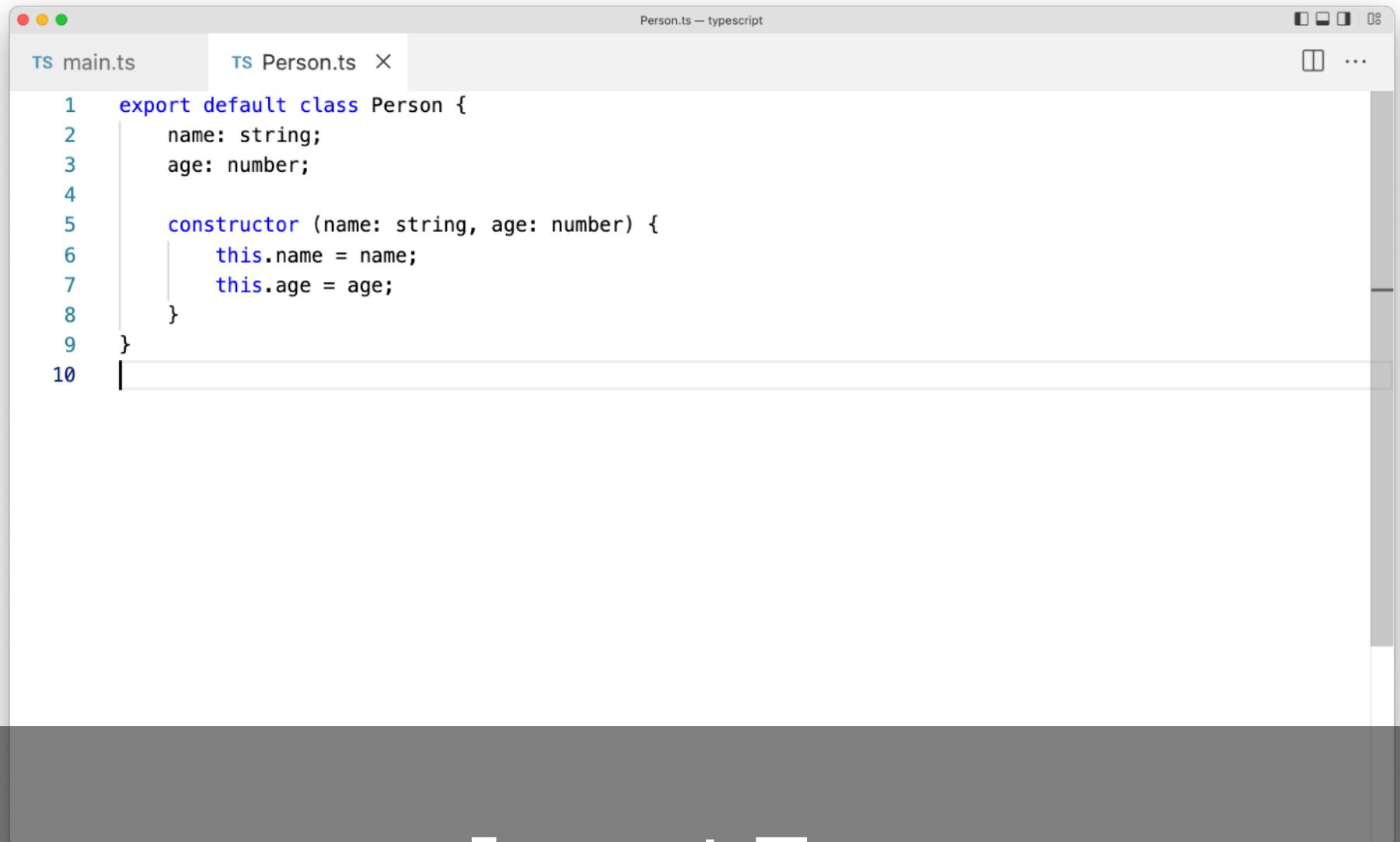
```
main.ts — typescript
TS main.ts    X  TS Repository.ts  TS Author.ts  TS Person.ts  ⋮  08

1 import Repository from "./Repository";
2 import Author from "./Author";
3
4 const authors = new Repository<Author>();
5 authors.add(new Author("Robert Martin", 62, [ { title: "Clean Code" }, { title: "Clean" }));
6 authors.add(new Author("Martin Fowler", 58, [ { title: "Refactoring" }]));
7
```

PROBLEMS OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

No problems have been detected in the workspace.

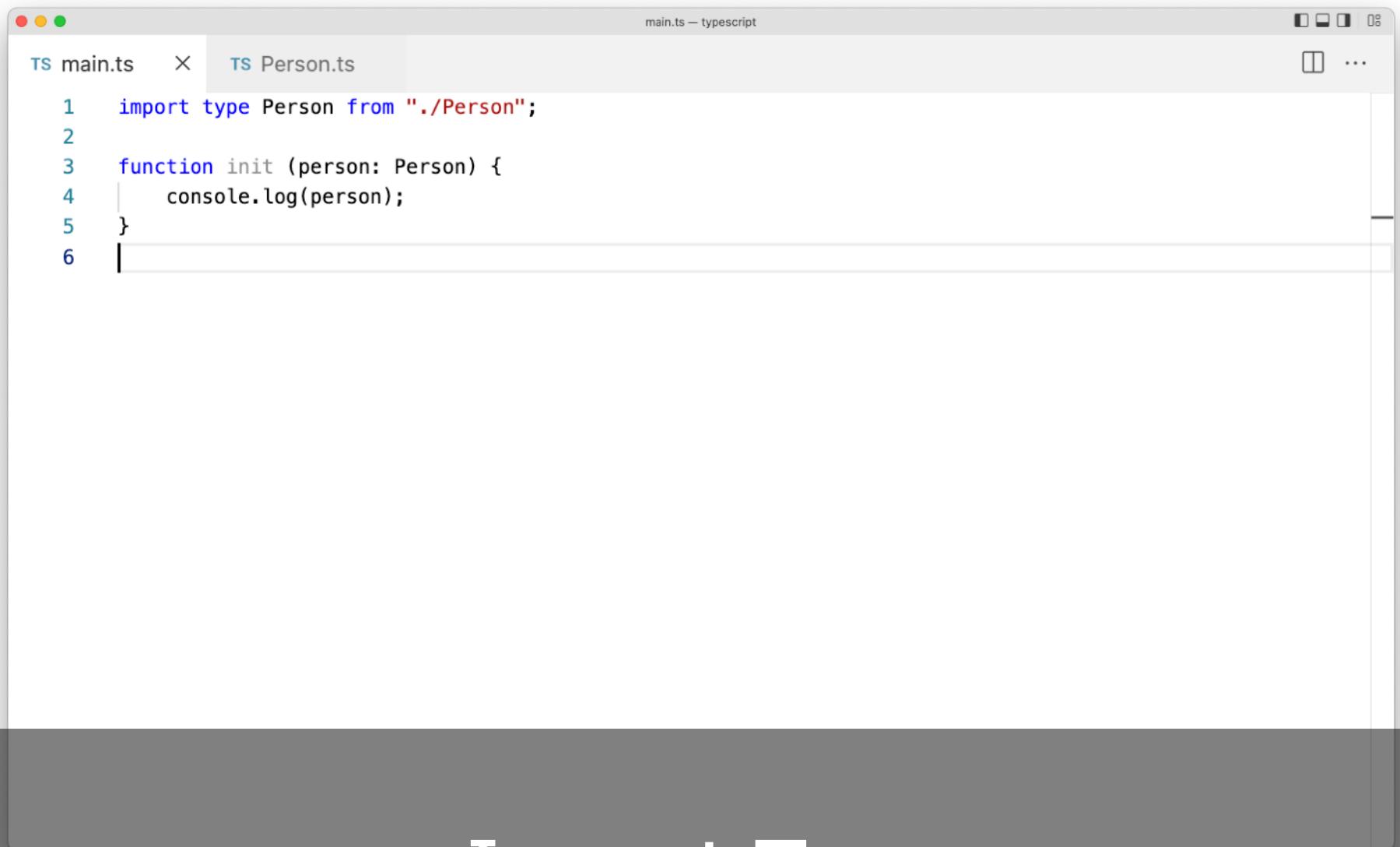
Exportando as classes por padrão



The screenshot shows a Mac OS X desktop environment with a window titled "Person.ts — typescript". Inside the window, there are two tabs: "main.ts" and "Person.ts". The "Person.ts" tab is active, displaying the following TypeScript code:

```
1  export default class Person {
2      name: string;
3      age: number;
4
5      constructor (name: string, age: number) {
6          this.name = name;
7          this.age = age;
8      }
9  }
10
```

Import Type



The screenshot shows a Mac OS X application window titled "main.ts — typescript". Inside the window, there are two tabs: "main.ts" and "Person.ts". The "main.ts" tab is active and contains the following code:

```
1 import type Person from "./Person";
2
3 function init (person: Person) {
4     console.log(person);
5 }
6
```

Import Type

The screenshot shows a Microsoft Visual Studio Code interface. The main editor window contains the following TypeScript code:

```
main.ts — typescript
TS main.ts 1 X TS Person.ts

1 import type Person from "./Person";
2
3 function init (person: Person) {
4     console.log(person);
5 }
6
7 init(new Person("Rodrigo Branas", 38));
8
```

The code uses the `import type` statement to import the `Person` type from `./Person`. The `Person` type is then used as a parameter type for the `init` function and as the value for the `new Person` constructor.

Below the editor, the **PROBLEMS** tab is selected, showing one error:

- TS main.ts src 1**
- > 💡 'Person' cannot be used as a value because it was imported using 'import type'. ts(1361) [Ln 7, Col 10]**

The error message indicates that the `Person` type, which was imported using `import type`, cannot be used as a value in the `new Person` expression. This is a TypeScript feature that prevents circular imports or incorrect type usage.

Import Type

Namespace

Um namespace é um basicamente um objeto que atua como uma coleção de variáveis, funções e classes, sendo importante para mediar conflitos de duplicação de nomes, principalmente em linguagens sem modularização

```
1 export namespace Vehicles {  
2     export class Bike {}  
3     export class Car {}  
4     export class Bus {}  
5     export class Truck {}  
6 }
```

PROBLEMS OUTPUT TERMINAL ... Filter (e.g. text, **/*.ts, !**/node_modules/**) ✖

No problems have been detected in the workspace.

namespace

The screenshot shows a code editor window titled "main.ts — typescript". The main editor area contains the following TypeScript code:

```
1 import { Vehicles } from "./Vehicles";
2
3 const bike = new Vehicles.Bike();
4 const car = new Vehicles.Car();
5 const bus = new Vehicles.Bus();
6 const truck = new Vehicles.Truck();
7
```

Below the editor, the "PROBLEMS" tab is selected, showing the message: "No problems have been detected in the workspace."

namespace



Qual é a diferença entre um module
um namespace?

A close-up photograph of a yellow caution tape. The word "CAUTION" is printed in large, bold, black capital letters. The tape is slightly curved, and the background shows some green foliage.

CAUTION

Como modules são padrão no ECMAScript,
utilize modules ao invés de namespaces

Enums

Com os enums é possível definir um conjunto de nomes padronizado e utilizado em uma variável ou propriedade de um objeto

A screenshot of a Mac OS X application window titled "Category.ts — typescript". The window has the standard red, yellow, and green close buttons in the top-left corner and a title bar with the file name and a "typescript" icon. In the top-right corner are standard window control icons. The main content area is a code editor with a light gray background. On the left side of the editor, there are line numbers from 1 to 6. The code itself is an export enum definition:

```
1  export enum Category {  
2      PROGRAMMING,  
3      SCIENCE,  
4      ENTERTAINMENT  
5  };  
6
```

The screenshot shows a code editor window with a light gray background. At the top, there's a title bar with three colored dots (red, yellow, green) on the left, the file name "main.ts — typescript" in the center, and standard window control icons (minimize, maximize, close) on the right.

The main area contains a code editor with the following content:

```
TS main.ts ×  
1 import { Category } from "./Category";  
2  
3 console.log(Category.PROGRAMMING);  
4 console.log(Category.SCIENCE);  
5 console.log(Category.ENTERTAINMENT);  
6
```

Below the code editor is a navigation bar with tabs: PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is underlined, indicating it is active. To the right of the tabs is a toolbar with icons for opening a new terminal ("zsh"), switching between terminals ("+"), closing the terminal ("X"), and other functions.

The terminal pane displays the following text:

```
rodrigobranas typescript $ npx ts-node src/main.ts  
0  
1  
2  
rodrigobranas typescript $ █
```

Category.ts — typescript

TS Category.ts X

```
1  export enum Category {  
2      PROGRAMMING = "Programming",  
3      SCIENCE = "Science",  
4      ENTERTAINMENT = "Entertainment"  
5  };  
6
```

The screenshot shows a code editor window with a tab bar at the top labeled "main.ts — typescript". The main area contains the following TypeScript code:

```
TS main.ts ×  
1 import { Category } from "./Category";  
2  
3 console.log(Category.PROGRAMMING);  
4 console.log(Category.SCIENCE);  
5 console.log(Category.ENTERTAINMENT);  
6
```

Below the code editor is a navigation bar with tabs: PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is active, indicated by a solid underline. To the right of the tabs is a toolbar with icons for opening a new terminal ("zsh"), switching between terminals ("+"), closing the terminal ("X"), and other functions.

The terminal output window displays the following text:

```
rodrigobranas typescript $ npx ts-node src/main.ts  
Programming  
Science  
Entertainment  
rodrigobranas typescript $ █
```

Author.ts — typescript

TS main.ts TS Author.ts X

```
3 export enum Category {  
4     PROGRAMMING,  
5     SCIENCE,  
6     ENTERTAINMENT  
7 }  
8  
9 export default class Author extends Person {  
10     constructor (name: string, age: number, readonly books: {  
11         title: string,  
12         category: Category  
13     }[]) {  
14         super(name, age);  
15     }  
16     getBooksCount(): number {  
17         return this.books.length;  
18     }  
19     addBook(book: {  
20         title: string;  
21         category: Category;  
22     }): void {  
23         this.books.push(book);  
24     }  
25     removeBook(index: number): void {  
26         this.books.splice(index, 1);  
27     }  
28 }  
29  
30 interface Person {  
31     name: string;  
32     age: number;  
33 }
```

PROBLEMS OUTPUT ... Filter (e.g. text, **/*.ts, !**/node_modules/**)

No problems have been detected in the workspace.

The screenshot shows a code editor interface with two tabs: "main.ts" and "Author.ts". The "main.ts" tab is active, displaying the following TypeScript code:

```
1 import Repository from "./Repository";
2 import Author, { Category } from "./Author";
3
4 const authors = new Repository<Author>();
5 authors.add(new Author("Robert Martin", 62, [
6     { title: "Clean Code", category: Category.PROGRAMMING },
7     { title: "Clean Architecture", category: Category.PROGRAMMING }]
8 )));
9 authors.add(new Author("Martin Fowler", 58, [
10    { title: "Refactoring", category: Category.PROGRAMMING }
11 )));
12
```

The "Author.ts" tab is visible in the background. Below the code editor is a "PROBLEMS" tab, which is currently selected, showing the message: "No problems have been detected in the workspace." To the right of the problems tab is a search bar labeled "Filter (e.g. text, **/*.ts, !**/node_modules/**)" with a magnifying glass icon.

Decorators

Os decorators são úteis para criar uma orientação à aspectos, de forma mais desacoplada, em classes, métodos, métodos de acesso, propriedades e parâmetros

A opção **experimentalDecorators** habilita a utilização de decorators

O **class decorator** é aplicado imediatamente antes da declaração de uma classe e permite interagir com a função construtora da classe

O **property decorator** é aplicado imediatamente antes da declaração de uma propriedade e permite interagir com a propriedade

O **method decorator** é aplicado imediatamente antes da declaração de um método e permite interagir com a assinatura do método