

Entendendo o Sistema de Tipos de TypeScript

Marcelo “Haskell” Camargo

github.com/haskellcamargo

marcelocamargo@linuxmail.org



Sumário

- 1) Introdução a TypeScript
- 2) Mergulhando no Sistema de Tipos

1 · Introdução a TypeScript

- 1) O que é TypeScript?
- 2) Quais as diferenças de TypeScript para JavaScript?
- 3) Quais as vantagens de TypeScript?

1 · 1 – 0 que é TypeScript?

- JavaScript como deveria ser
- Superset de ECMAScript
- Sistema de tipos decente
- Criado pela Microsoft
- 100% open-source


1 · 2 – Quais as diferenças de TS para JS?

- Sistema de tipos com checagem em tempo de compilação!

1 · 3 – Quais as vantagens de TypeScript?

- Coerência de tipos e de dados
- Captura de erros em tempo de compilação
- Features de ECMAScript 6 (código retrocompatível!)
- Linter com *code-completion*
- Sistema de módulos padrão
- Abstração e expressividade

1 · 3 · A – Coerência em tipos de dados



The screenshot shows a code editor with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar with three tabs: Main.ts, App.hs, and Mapper.urano. The code in Main.ts is as follows:

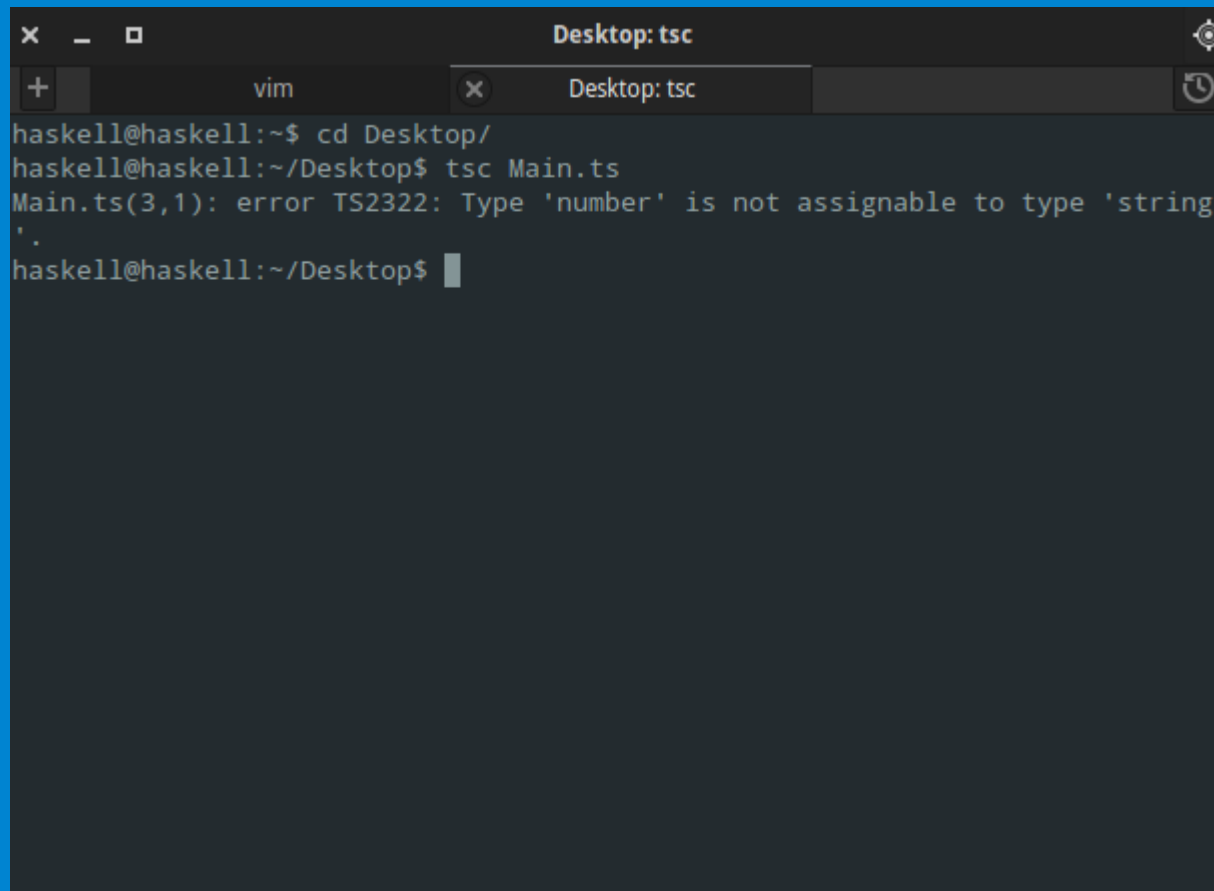
```
1  
2 var x: string = "A";  
3 x = 10;  
4
```

The third line, `x = 10;`, is highlighted with a light gray background. A red squiggly line is under the variable `x`, and a red error message is displayed at the bottom of the editor:

Type 'number' is not assignable to type 'string'. WakaTime active 12:59 AM, Line 3, Column 2

The status bar at the bottom right shows "Spaces: 2" and "TypeScript".

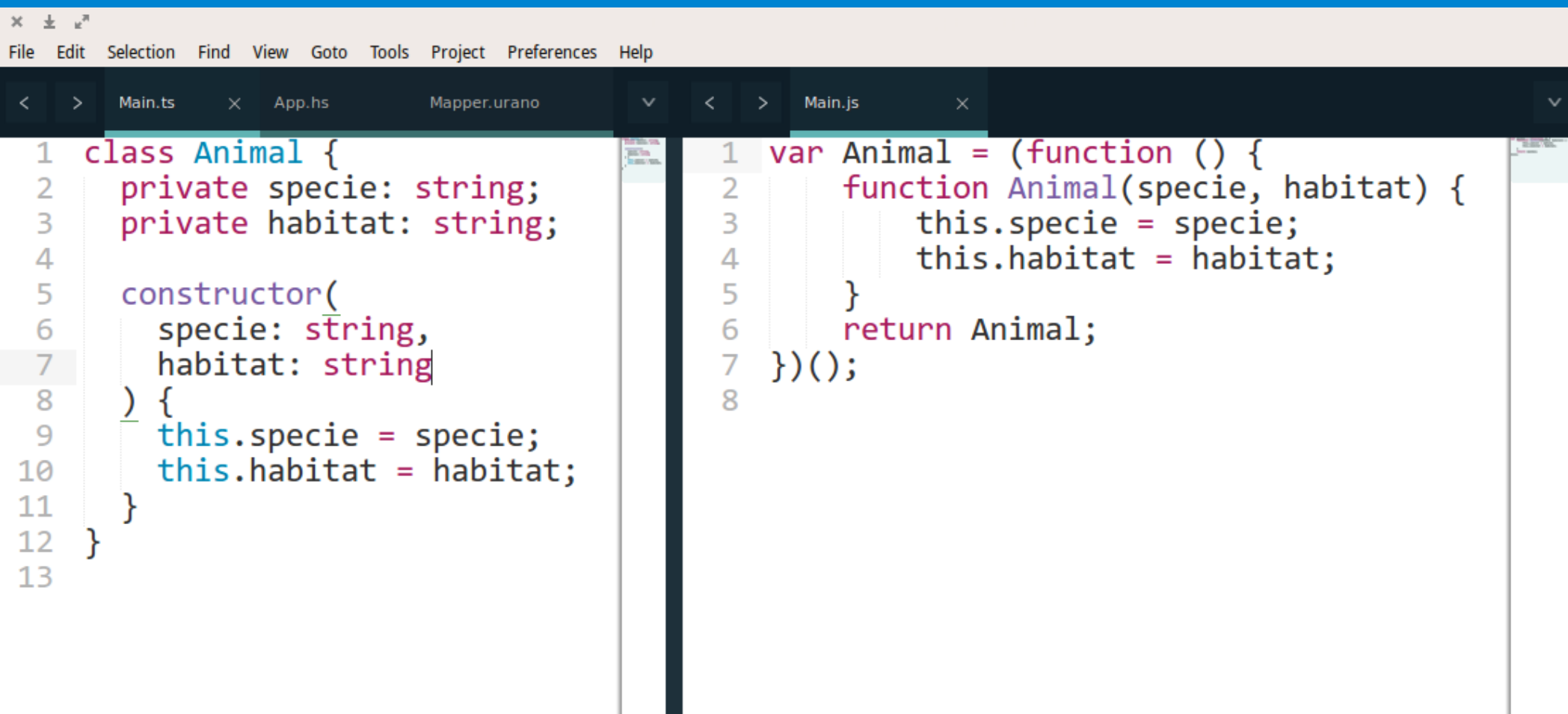
1 · 3 · B — Capturas de erros em tempo de compilação



```
haskell@haskell:~$ cd Desktop/  
haskell@haskell:~/Desktop$ tsc Main.ts  
Main.ts(3,1): error TS2322: Type 'number' is not assignable to type 'string'.  
haskell@haskell:~/Desktop$
```

The image shows a terminal window titled "Desktop: tsc". The window has a dark background and a light-colored text. The terminal output shows the user navigating to the Desktop directory and running the TypeScript compiler (tsc) on Main.ts. The compiler reports an error: "Main.ts(3,1): error TS2322: Type 'number' is not assignable to type 'string'." The user's prompt is "haskell@haskell:~/Desktop\$".

1 · 3 · C – Features de ECMAScript 6

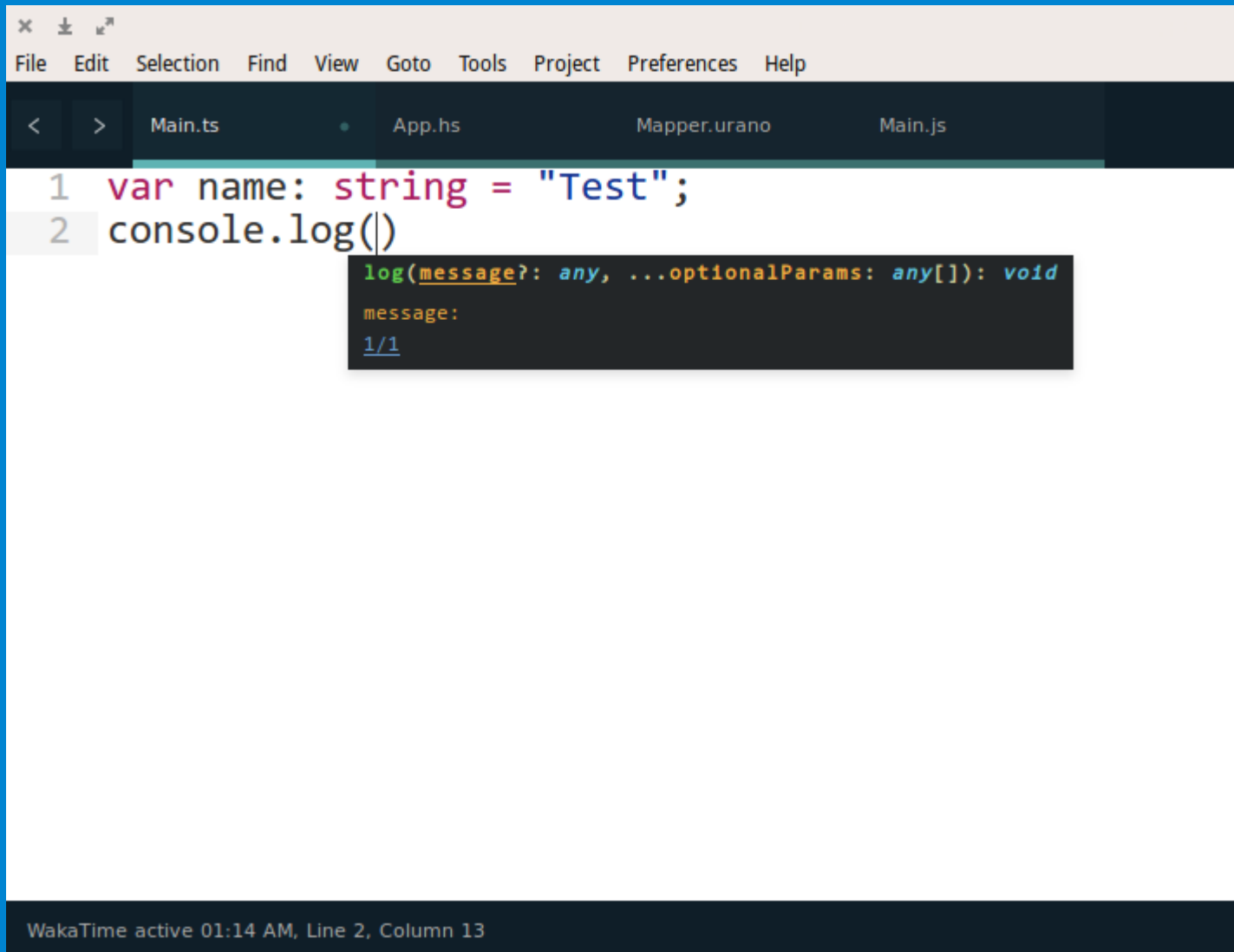


The image shows a side-by-side comparison of two code snippets in a code editor. The left pane, titled 'Main.ts', shows a TypeScript class definition for 'Animal' using ES6 class syntax. It includes private fields 'specie' and 'habitat' of type 'string', and a constructor that takes 'specie' and 'habitat' as arguments and assigns them to 'this.specie' and 'this.habitat'. The right pane, titled 'Main.js', shows the equivalent JavaScript code using a function constructor pattern. It defines a function 'Animal' that takes 'specie' and 'habitat' as arguments and returns a new object with 'specie' and 'habitat' properties. The code is color-coded: keywords like 'class', 'function', 'var', 'return', 'this', and 'constructor' are in blue or purple, while string literals and variable names are in black or grey.

```
1 class Animal {  
2   private specie: string;  
3   private habitat: string;  
4  
5   constructor(  
6     specie: string,  
7     habitat: string  
8   ) {  
9     this.specie = specie;  
10    this.habitat = habitat;  
11  }  
12 }  
13
```

```
1 var Animal = (function () {  
2   function Animal(specie, habitat) {  
3     this.specie = specie;  
4     this.habitat = habitat;  
5   }  
6   return Animal;  
7 })();  
8
```

1 · 3 · D – Linter com *code-completion*



The screenshot shows an IDE window with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar with four tabs: Main.ts, App.hs, Mapper.urano, and Main.js. The Main.ts tab is active, showing two lines of code: `1 var name: string = "Test";` and `2 console.log(|`. A dropdown menu is visible below the second line, showing the signature `log(message?: any, ...optionalParams: any[]): void` and the parameter `message:` with a link `1/1` below it. The status bar at the bottom indicates "WakaTime active 01:14 AM, Line 2, Column 13".

```
1 var name: string = "Test";
2 console.log(|
    log(message?: any, ...optionalParams: any[]): void
    message:
    1/1
```

WakaTime active 01:14 AM, Line 2, Column 13

1 · 3 · E – Sistema de módulos padrão

```
1 module Prelude {  
2   function map<T>(fn: (x: T) => T, xs: T[]): T[] {  
3     return xs.length  
4       ? map<T>(fn, [fn(xs[0])].concat(xs.slice(1)))  
5       : xs;  
6   }  
7 }  
8
```

```
1 var Prelude;  
2 (function (Prelude) {  
3   function map(fn, xs) {  
4     return xs.length  
5       ? map(fn, [fn(xs[0])].concat(xs.slice(1)))  
6       : xs;  
7   }  
8 })(Prelude || (Prelude = {}));  
9
```

WakaTime active 01:21 AM, Line 8, Column 9

Spaces: 2 JavaScript

1 · 3 · F – Abstração e expressividade

WTF!?

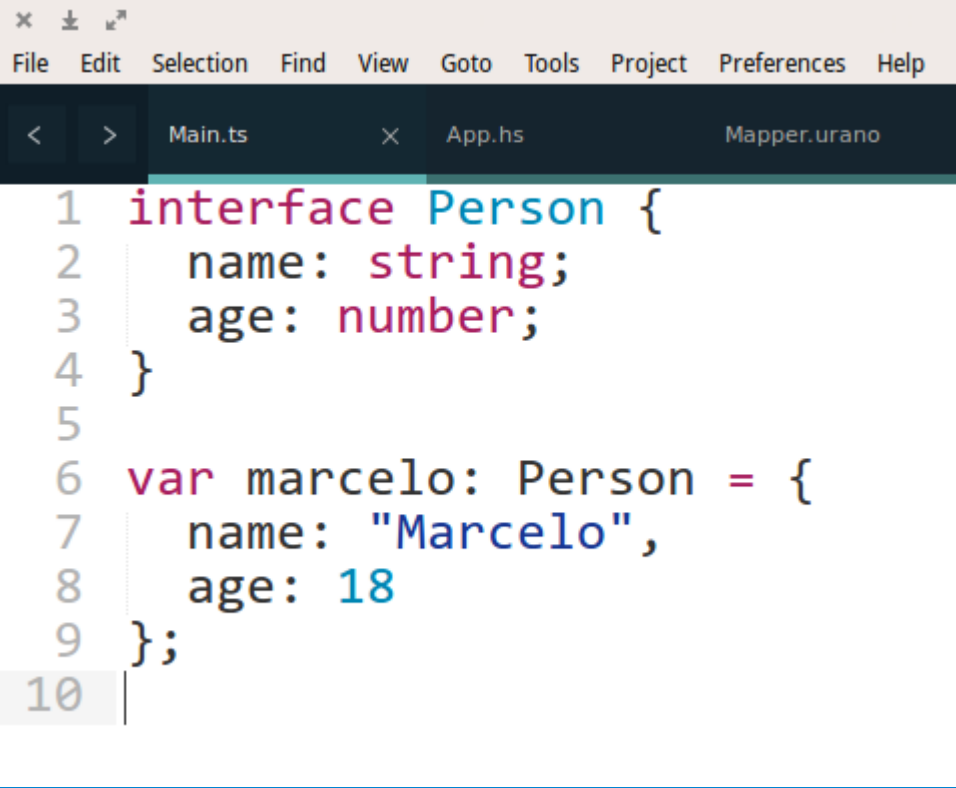
2 · Mergulhando no Sistema de Tipos

- 1) Tipos básicos
- 2) Interfaces
- 3) Function types
- 4) Array types
- 5) Class types
- 6) Extensão de interfaces
- 7) Tipos híbridos
- 8) Type-aliases
- 9) Extensão de tipos nativos
- 10) Tipos genéricos
- 11) Misturando tipos

2 · 1 – Tipos básicos

```
< > Main.ts × App.hs Mapper.urano Mair
1 var isDone: boolean = false;
2 var height: number = 6;
3 var name: string = "Bob";
4 var ages: number[] = [1, 2, 3];
5 enum Status {
6     Play,
7     Pause
8 }
9 var yourBelly: any = "-\\_(ツ)_/^-";
10 var x: void; // Why!?
11 |
```

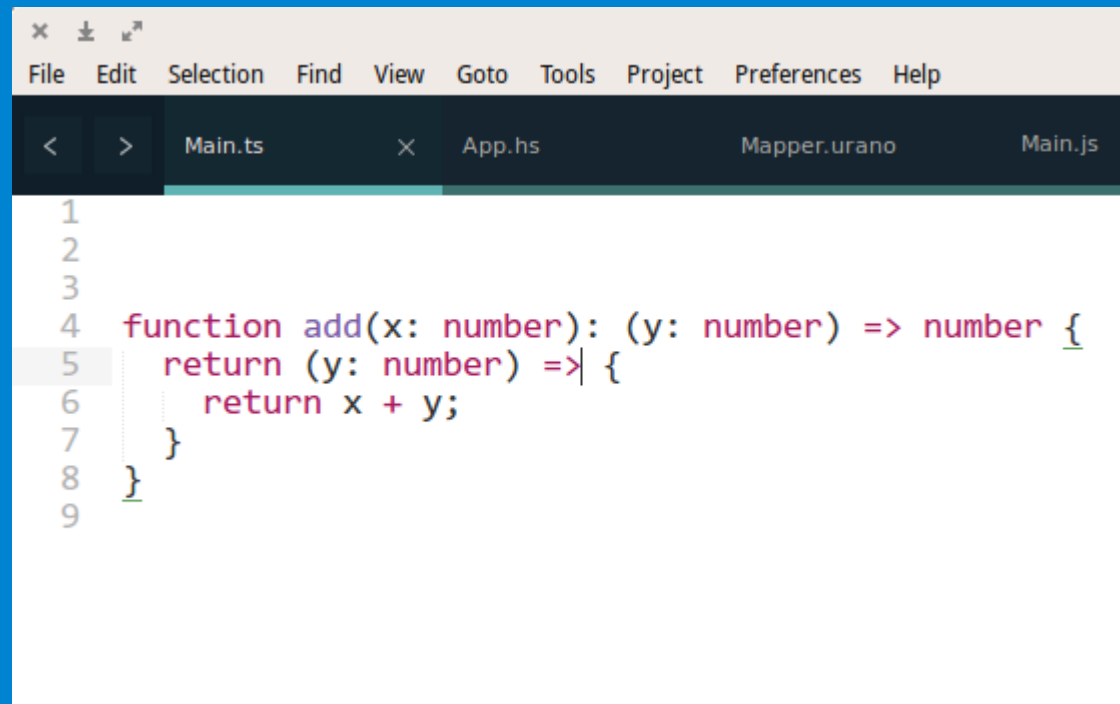
2 · 2 – Interfaces



The screenshot shows a code editor with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar with three tabs: Main.ts, App.hs, and Mapper.urano. The code in Main.ts defines an interface and a variable:

```
1 interface Person {  
2     name: string;  
3     age: number;  
4 }  
5  
6 var marcelo: Person = {  
7     name: "Marcelo",  
8     age: 18  
9 };  
10
```

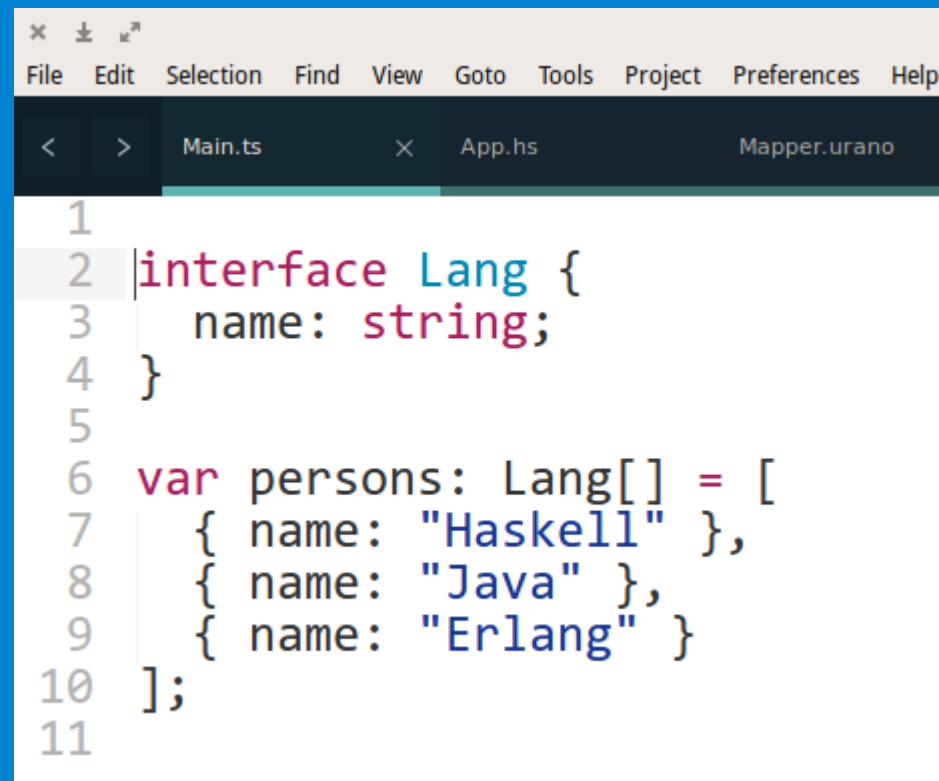
2 · 3 – Function types



A screenshot of a code editor window with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar showing four files: Main.ts, App.hs, Mapper.urano, and Main.js. The Main.ts file is active, displaying a TypeScript function definition. The code is as follows:

```
1  
2  
3  
4 function add(x: number): (y: number) => number {  
5   return (y: number) => {  
6     return x + y;  
7   }  
8 }  
9
```

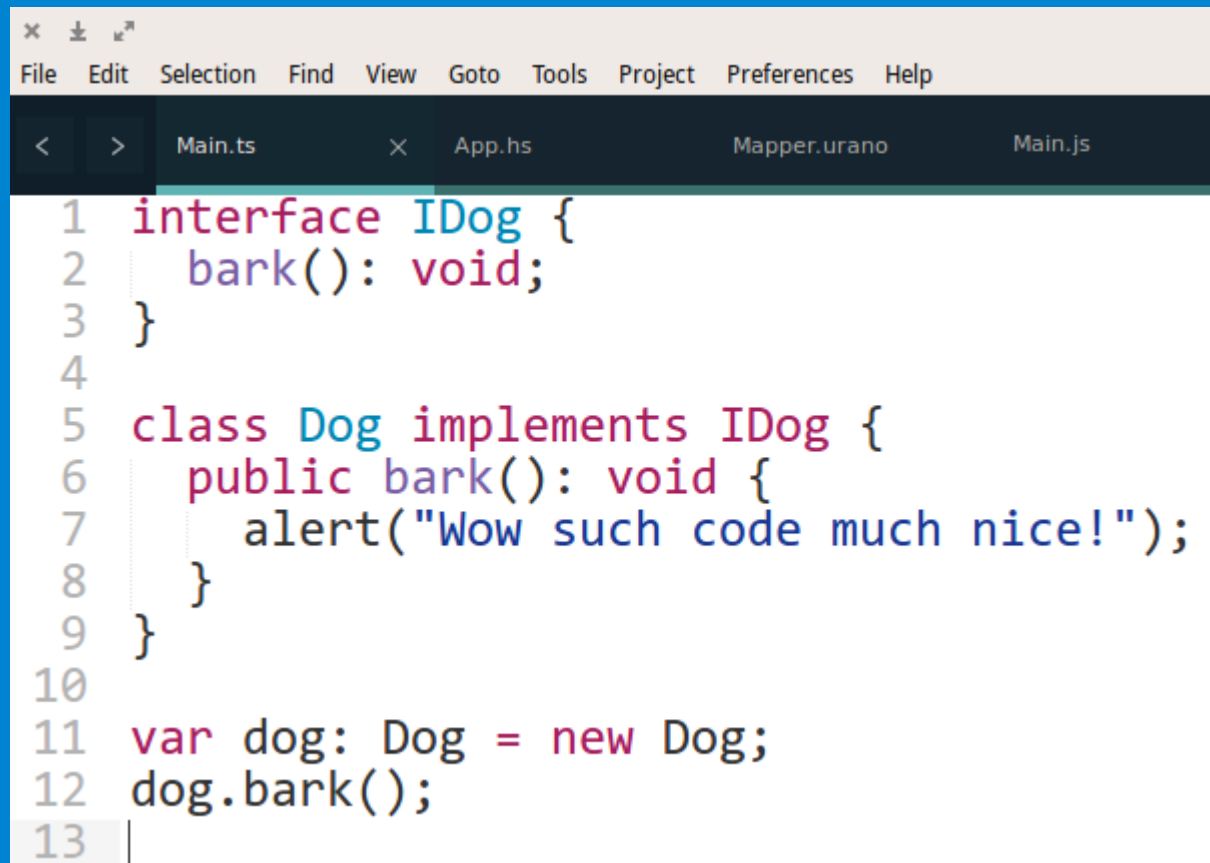

2 · 4 – Array types



A screenshot of a code editor window with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar showing three files: Main.ts, App.hs, and Mapper.urano. The Main.ts file is active, displaying the following TypeScript code:

```
1
2 interface Lang {
3     name: string;
4 }
5
6 var persons: Lang[] = [
7     { name: "Haskell" },
8     { name: "Java" },
9     { name: "Erlang" }
10 ];
11
```

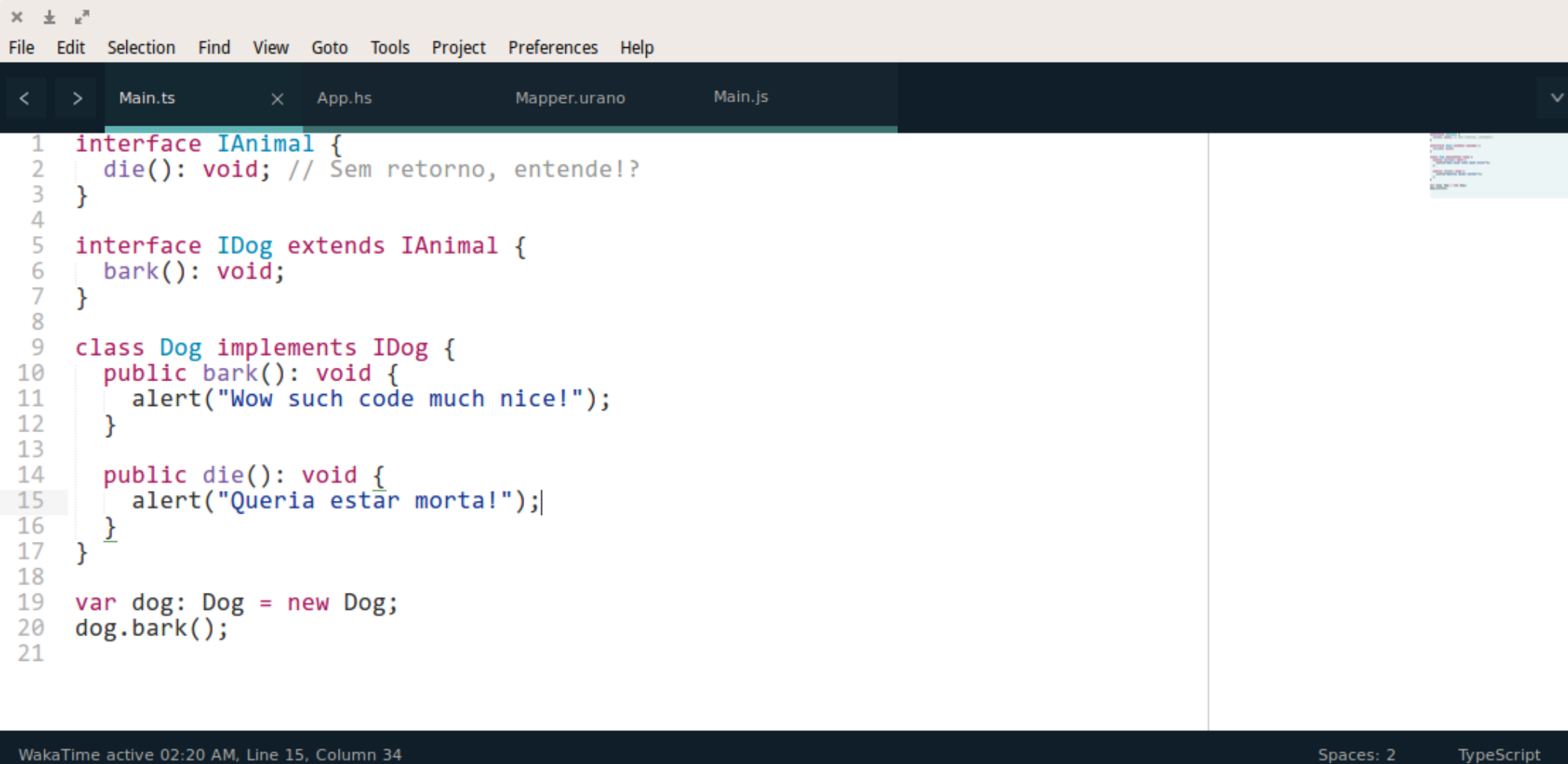
2 · 5 – Class types



The screenshot shows a code editor with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar with four tabs: Main.ts, App.hs, Mapper.urano, and Main.js. The Main.ts tab is active, displaying the following TypeScript code:

```
1 interface IDog {  
2     bark(): void;  
3 }  
4  
5 class Dog implements IDog {  
6     public bark(): void {  
7         alert("Wow such code much nice!");  
8     }  
9 }  
10  
11 var dog: Dog = new Dog;  
12 dog.bark();  
13
```

2 · 6 – Extensão de interfaces



The screenshot shows a code editor with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar with four tabs: Main.ts, App.hs, Mapper.urano, and Main.js. The Main.ts tab is active, displaying the following TypeScript code:

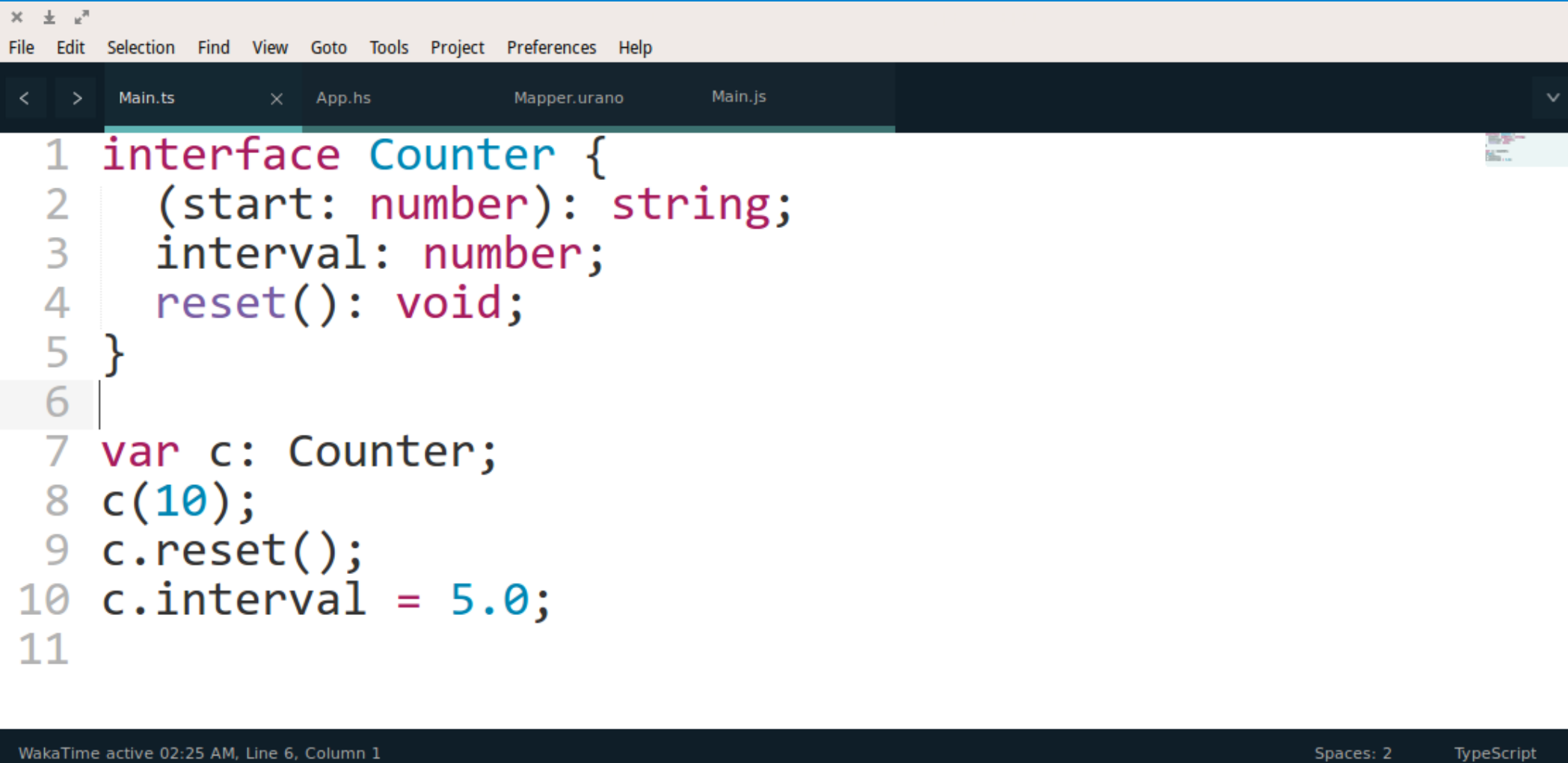
```
1 interface IAnimal {
2     die(): void; // Sem retorno, entende!?
3 }
4
5 interface IDog extends IAnimal {
6     bark(): void;
7 }
8
9 class Dog implements IDog {
10     public bark(): void {
11         alert("Wow such code much nice!");
12     }
13
14     public die(): void {
15         alert("Queria estar morta!");
16     }
17 }
18
19 var dog: Dog = new Dog;
20 dog.bark();
21
```

The code defines an `IAnimal` interface with a `die()` method, an `IDog` interface that extends `IAnimal` with a `bark()` method, and a `Dog` class that implements `IDog`. The `Dog` class has two methods: `bark()` and `die()`. The `die()` method is highlighted in the original image.

WakaTime active 02:20 AM, Line 15, Column 34

Spaces: 2 TypeScript

2 · 7 – Tipos híbridos

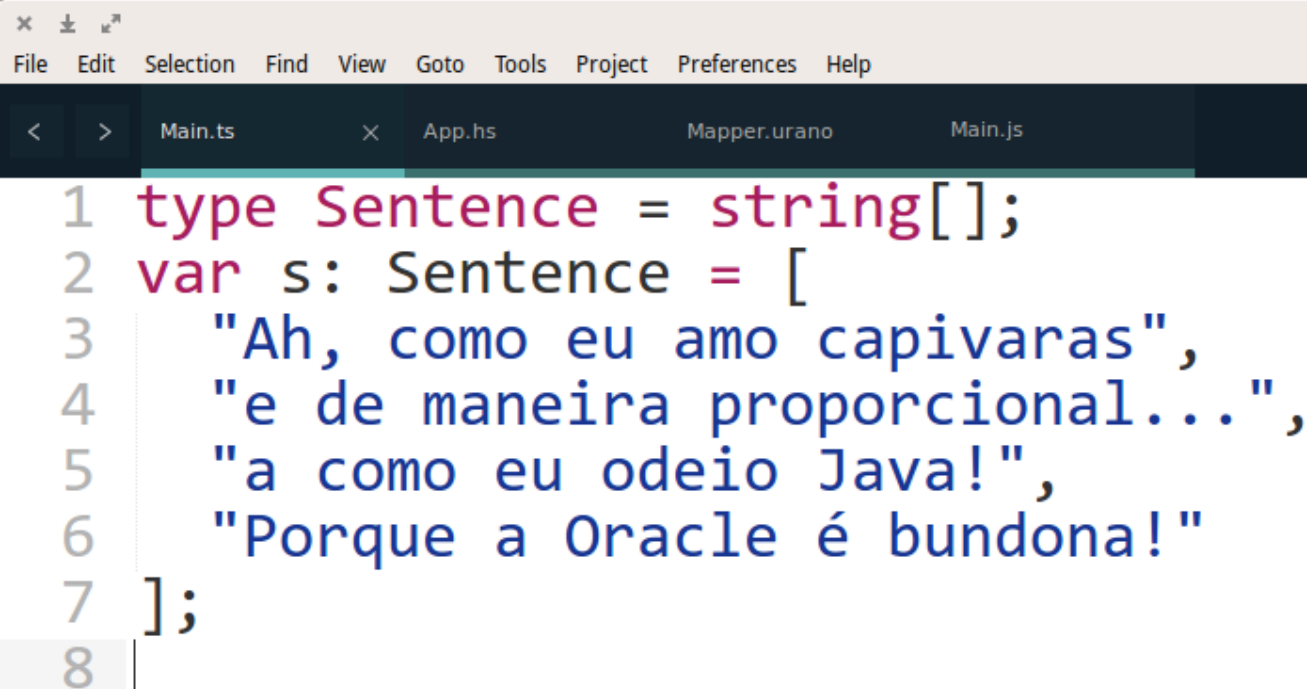


```
1 interface Counter {
2   (start: number): string;
3   interval: number;
4   reset(): void;
5 }
6
7 var c: Counter;
8 c(10);
9 c.reset();
10 c.interval = 5.0;
11
```

WakaTime active 02:25 AM, Line 6, Column 1

Spaces: 2 TypeScript

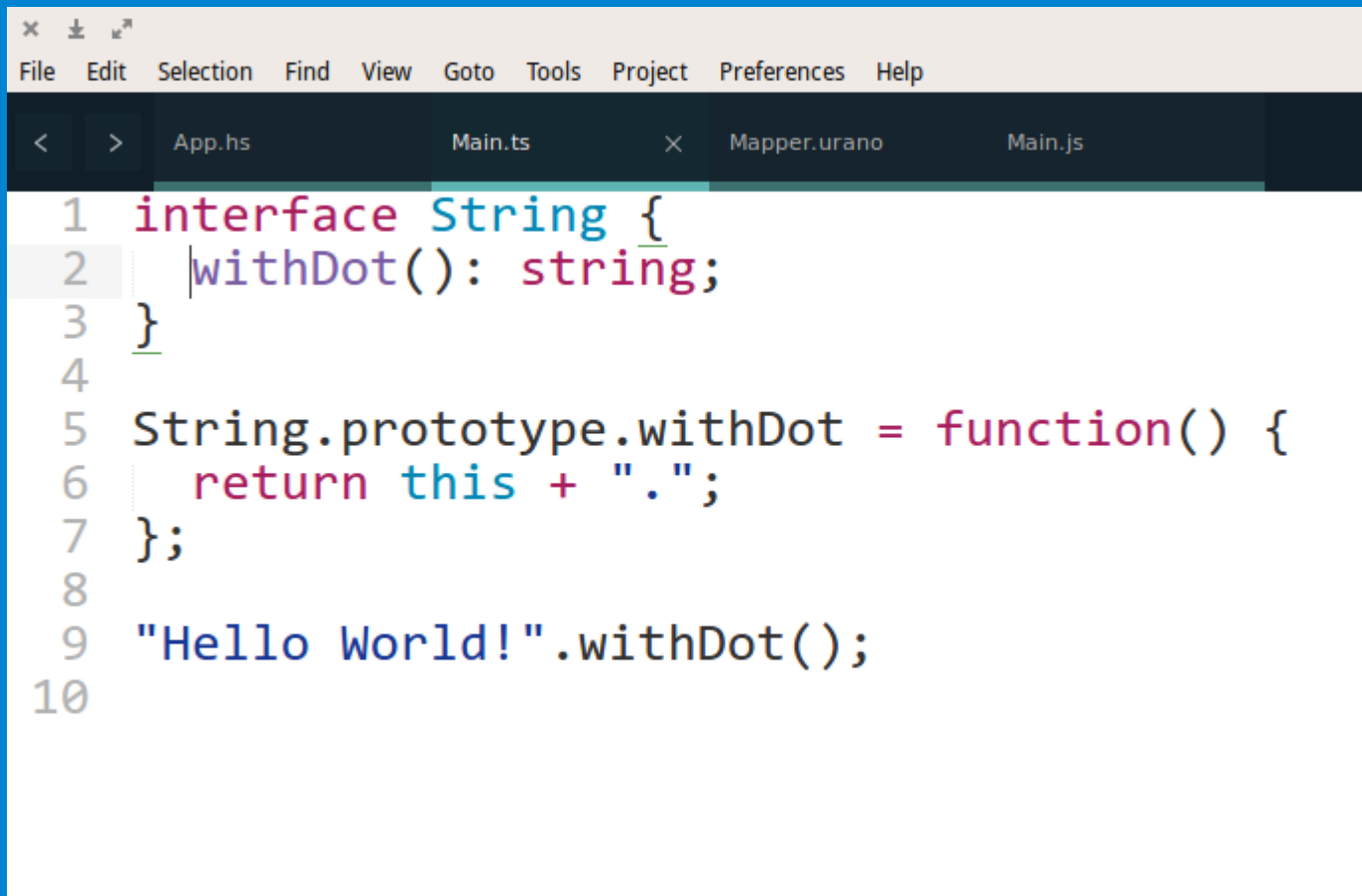
2 · 8 – *Type-alias*



The screenshot shows a code editor with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar with four tabs: Main.ts, App.hs, Mapper.urano, and Main.js. The active tab is Main.ts, which contains the following TypeScript code:

```
1 type Sentence = string[];
2 var s: Sentence = [
3     "Ah, como eu amo capivaras",
4     "e de maneira proporcional...",
5     "a como eu odeio Java!",
6     "Porque a Oracle é bundona!"
7 ];
8
```

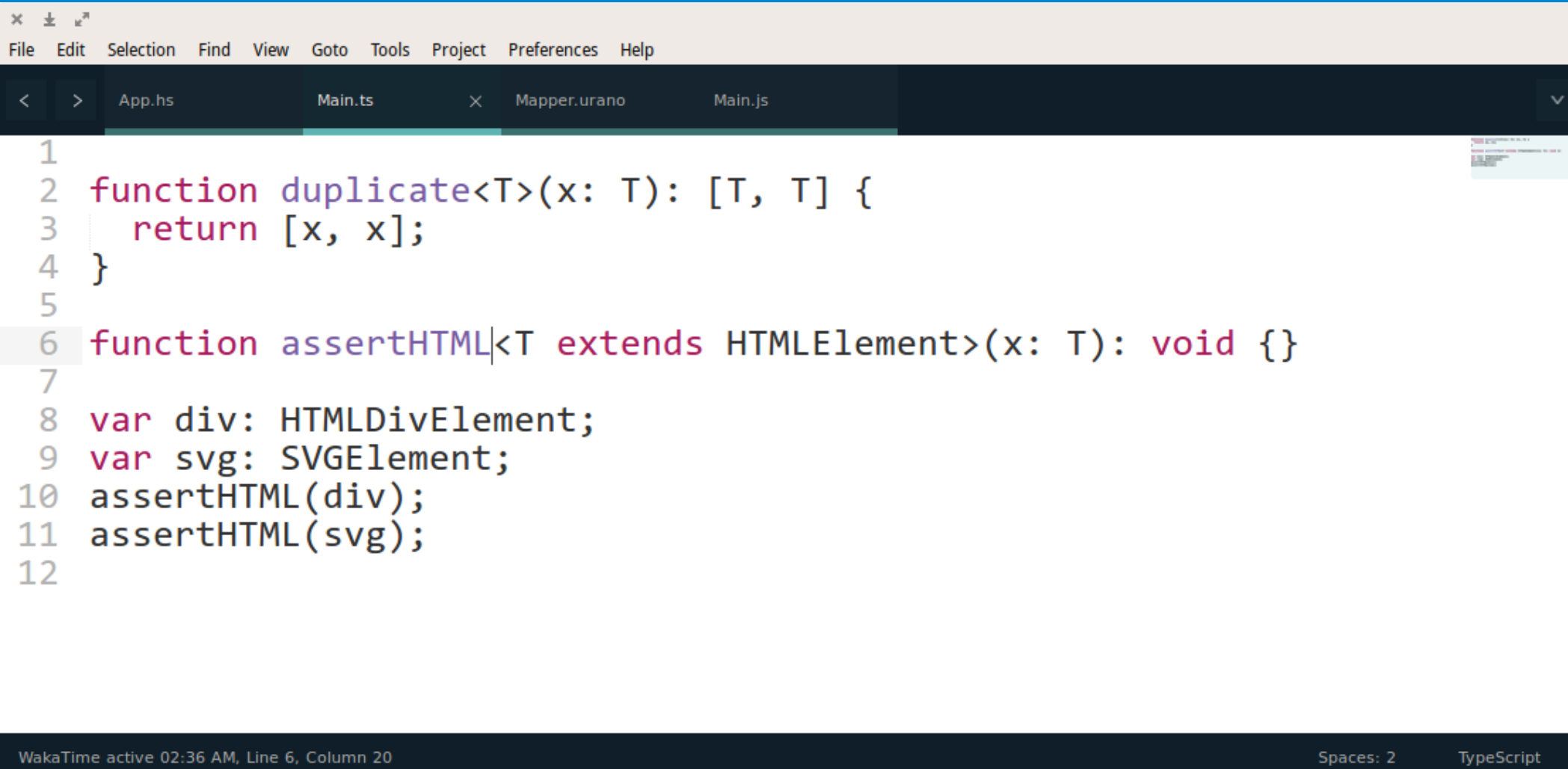
2 · 9 – Extensão de tipos nativos



```
1 interface String {  
2   |withDot(): string;  
3 }  
4  
5 String.prototype.withDot = function() {  
6   |return this + ".";  
7 };  
8  
9 "Hello World!".withDot();  
10
```

The image shows a code editor window with a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a tab bar with files App.hs, Main.ts, Mapper.urano, and Main.js. The code in Main.ts defines an interface String with a withDot() method, extends the String prototype with a withDot function, and then calls the method on the string "Hello World!".

2 · 10 – Tipos genéricos

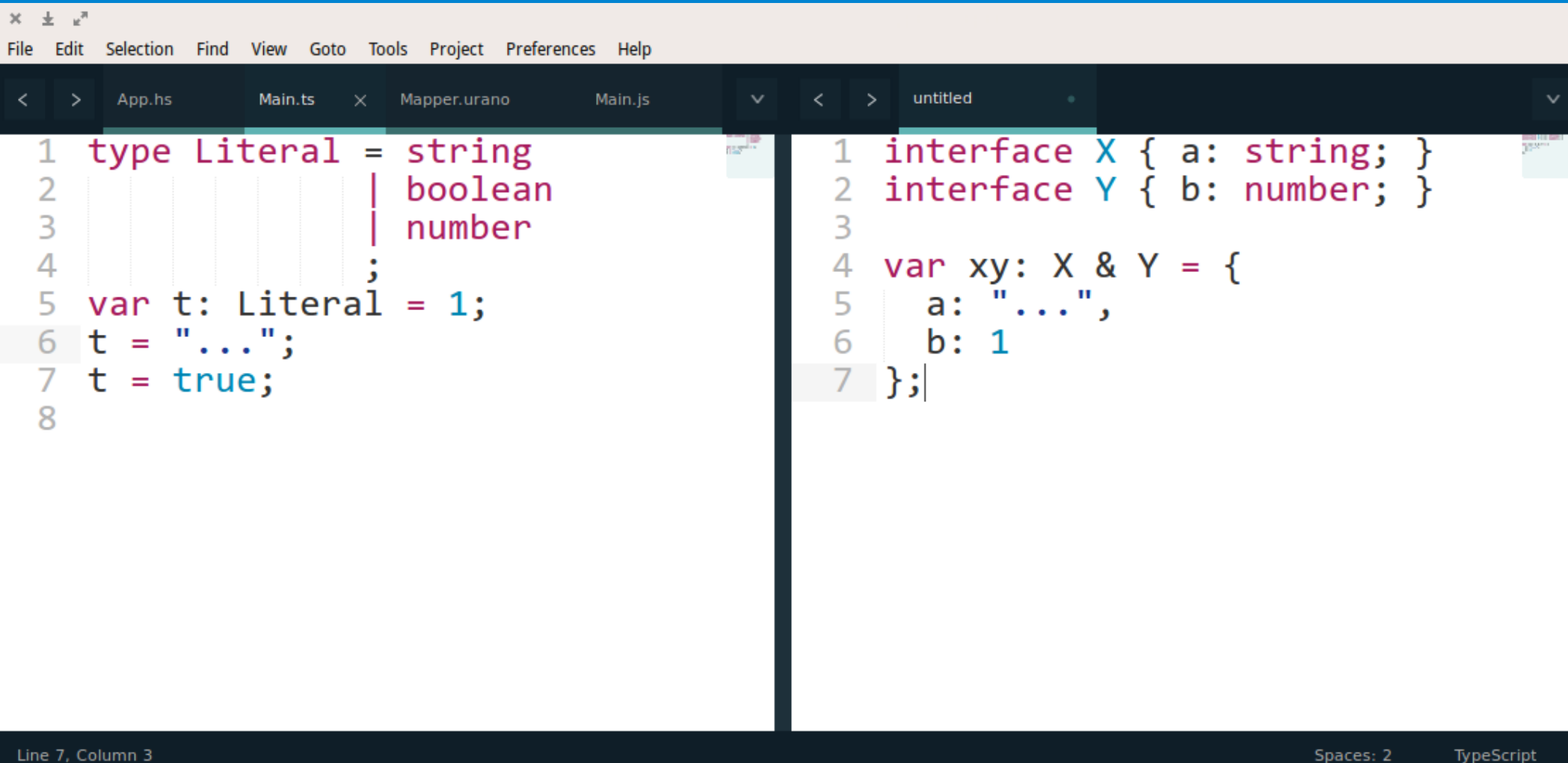


```
1  
2 function duplicate<T>(x: T): [T, T] {  
3     return [x, x];  
4 }  
5  
6 function assertHTML<T extends HTMLElement>(x: T): void {}  
7  
8 var div: HTMLDivElement;  
9 var svg: SVGElement;  
10 assertHTML(div);  
11 assertHTML(svg);  
12
```

WakaTime active 02:36 AM, Line 6, Column 20

Spaces: 2 TypeScript

2 · 11 – Misturando tipos



The image shows a screenshot of a code editor with two panels. The left panel displays a TypeScript file named 'Main.ts' with the following code:

```
1 type Literal = string
2               | boolean
3               | number
4               ;
5 var t: Literal = 1;
6 t = "...";
7 t = true;
8
```

The right panel displays an 'untitled' TypeScript file with the following code:

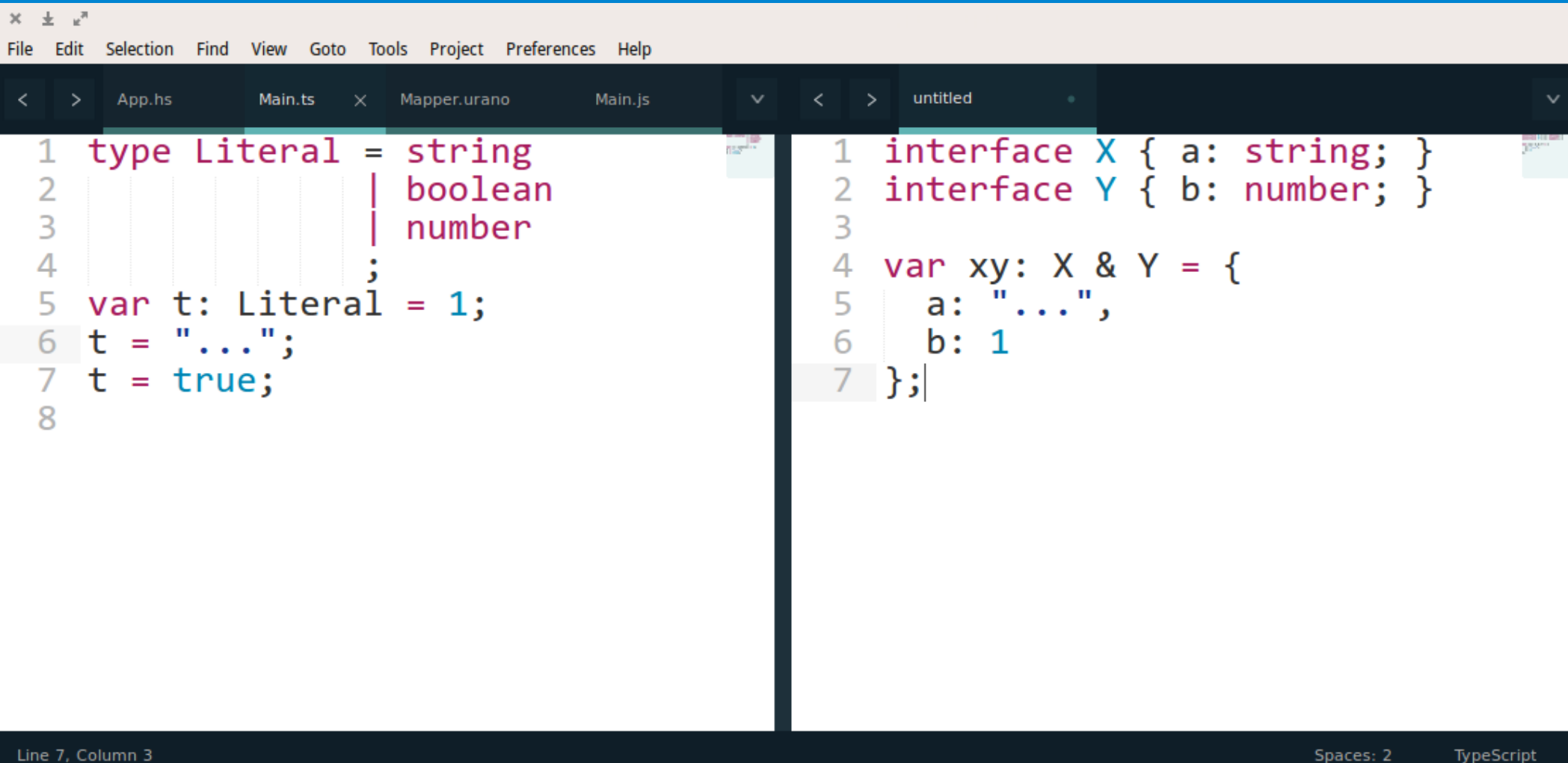
```
1 interface X { a: string; }
2 interface Y { b: number; }
3
4 var xy: X & Y = {
5     a: "...",
6     b: 1
7 };|
```

The status bar at the bottom indicates 'Line 7, Column 3' on the left and 'Spaces: 2 TypeScript' on the right.

“Capivaras são amigas, não comida”
Perguntas?



2 · 11 – Misturando tipos



The screenshot shows a code editor with two files open. The left file, 'Main.ts', contains a union type 'Literal' and a variable 't' of that type. The right file, 'untitled', contains two interfaces 'X' and 'Y', and a variable 'xy' of their intersection type 'X & Y'.

```
1 type Literal = string
2               | boolean
3               | number
4               ;
5 var t: Literal = 1;
6 t = "...";
7 t = true;
8
```

```
1 interface X { a: string; }
2 interface Y { b: number; }
3
4 var xy: X & Y = {
5     a: "...",
6     b: 1
7 };|
```

Line 7, Column 3

Spaces: 2 TypeScript