We don't want to migrate to TypeScript, there is too much to learn!

Real Talk

Who develops websites or codes for the JavaScript ecosystem?

Who writes JavaScript without TypeScript?

Why?

Too hard / Complicated

true if you tried it in the early days
like version 2.0

Angular was the issue (more about that later)

No need

I thought that, but today, even my littlest scripts are done with TS

Too slow to work with?

At the beginning, I agreed
But with experience, the DX it too great:

- autocomplete
- type check
- great for team work
- great for documentation
- ...

Compiler performances

I agree that `tsc` is not the best in performances. Some people gave a shot at Go and Rust to write an improved `tsc`, but no release at the moment

Tired of squiggly red lines

We’ll see together how to migrate slowly

Lead Frontend Developer, Streamer at BearStudio

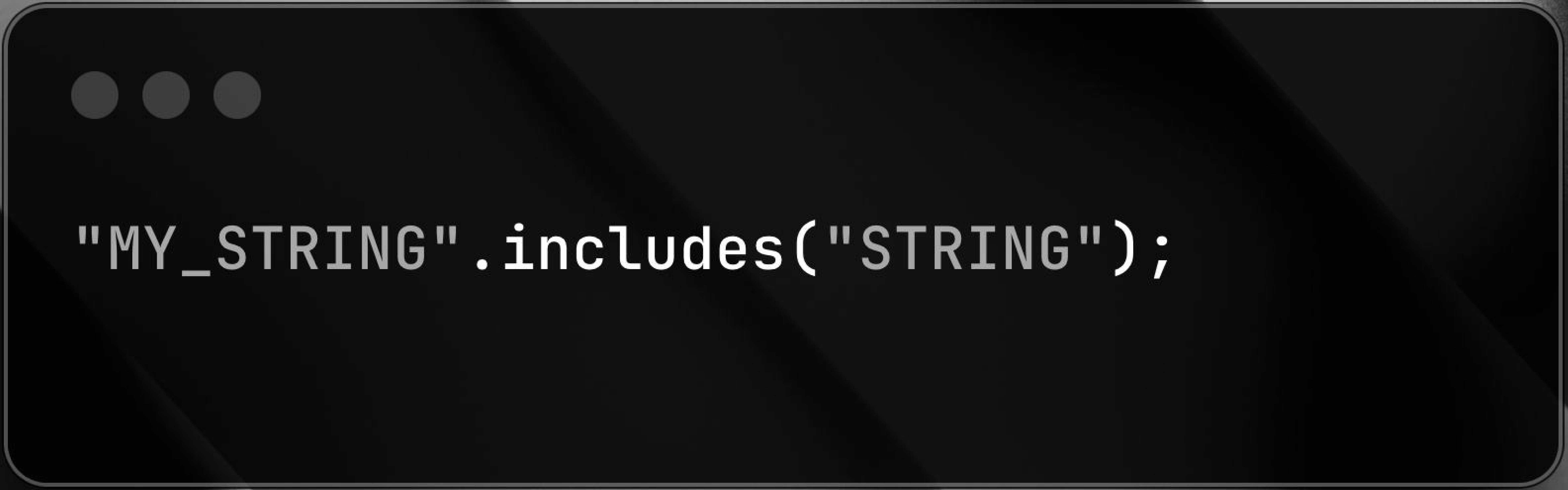
Yoann Fleury

- ⚛️ React and 🔧 TypeScript expert
- 🌍 Rouen, France
- 🗣️ @yoannfleurydev on socials

Write JavaScript

Enjoy types without TypeScript

With the right tools



```
"MY_STRING".includes("STRING");
```

```
"MY_STRING".repeat(2);
```

```
"MY_STRING".toLowercase();
```

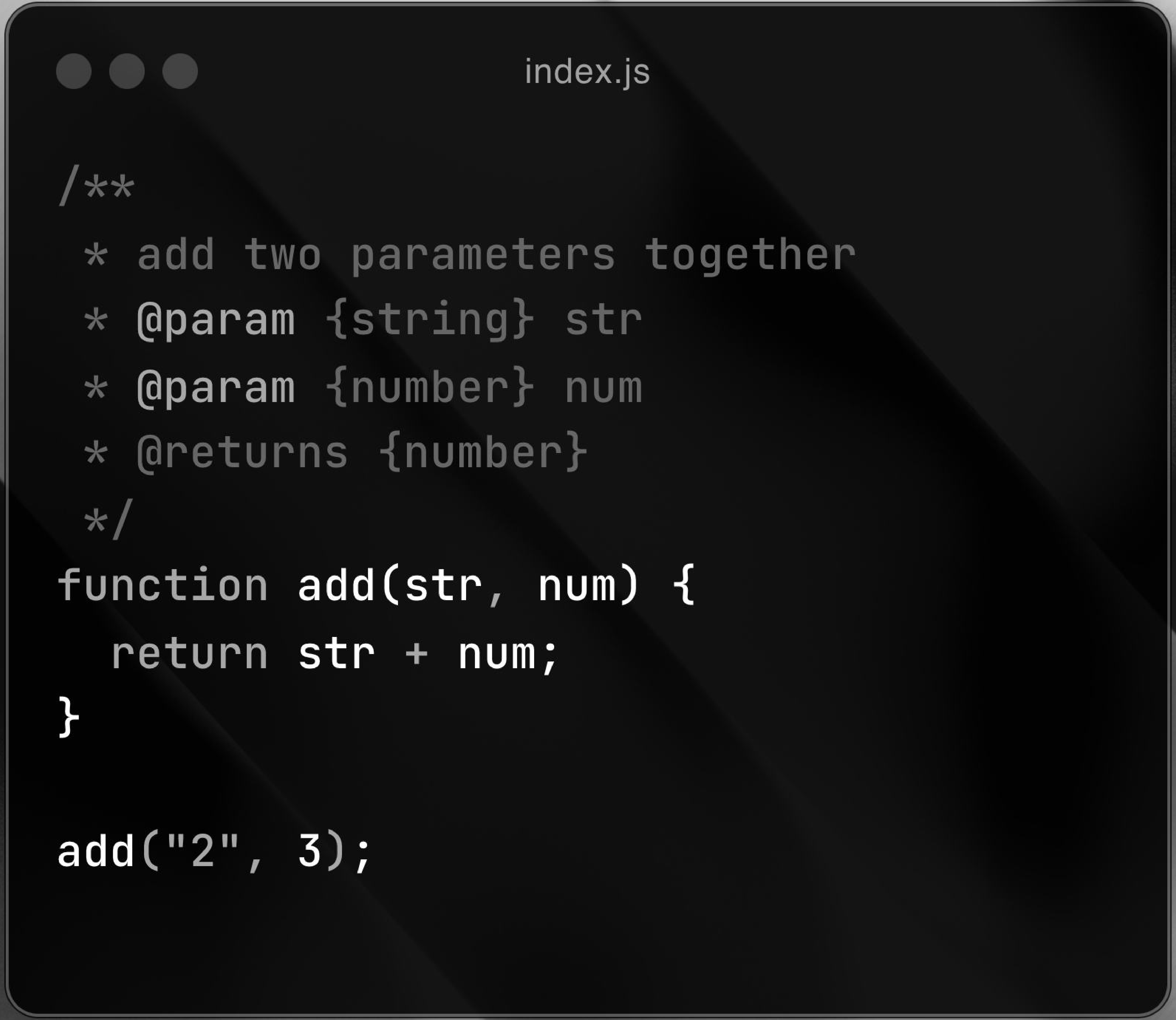
```
// @ts-check
```

JS 01.tscheck.2.js

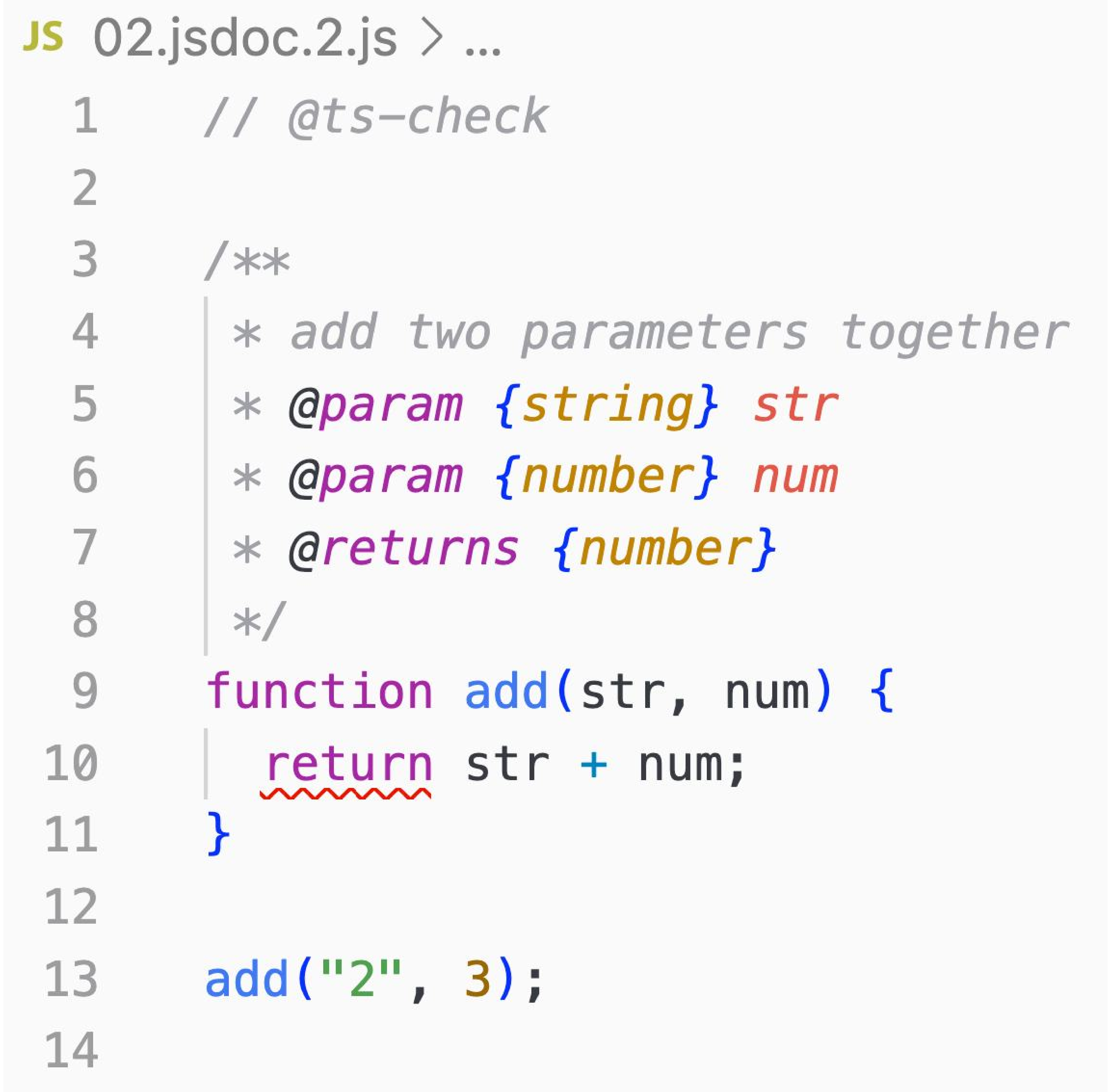
```
// @ts-check

"MY_STRING".toLowercase();
```

```
"MY_STRING".toLowerCase();
                 ^
```

JSDoc

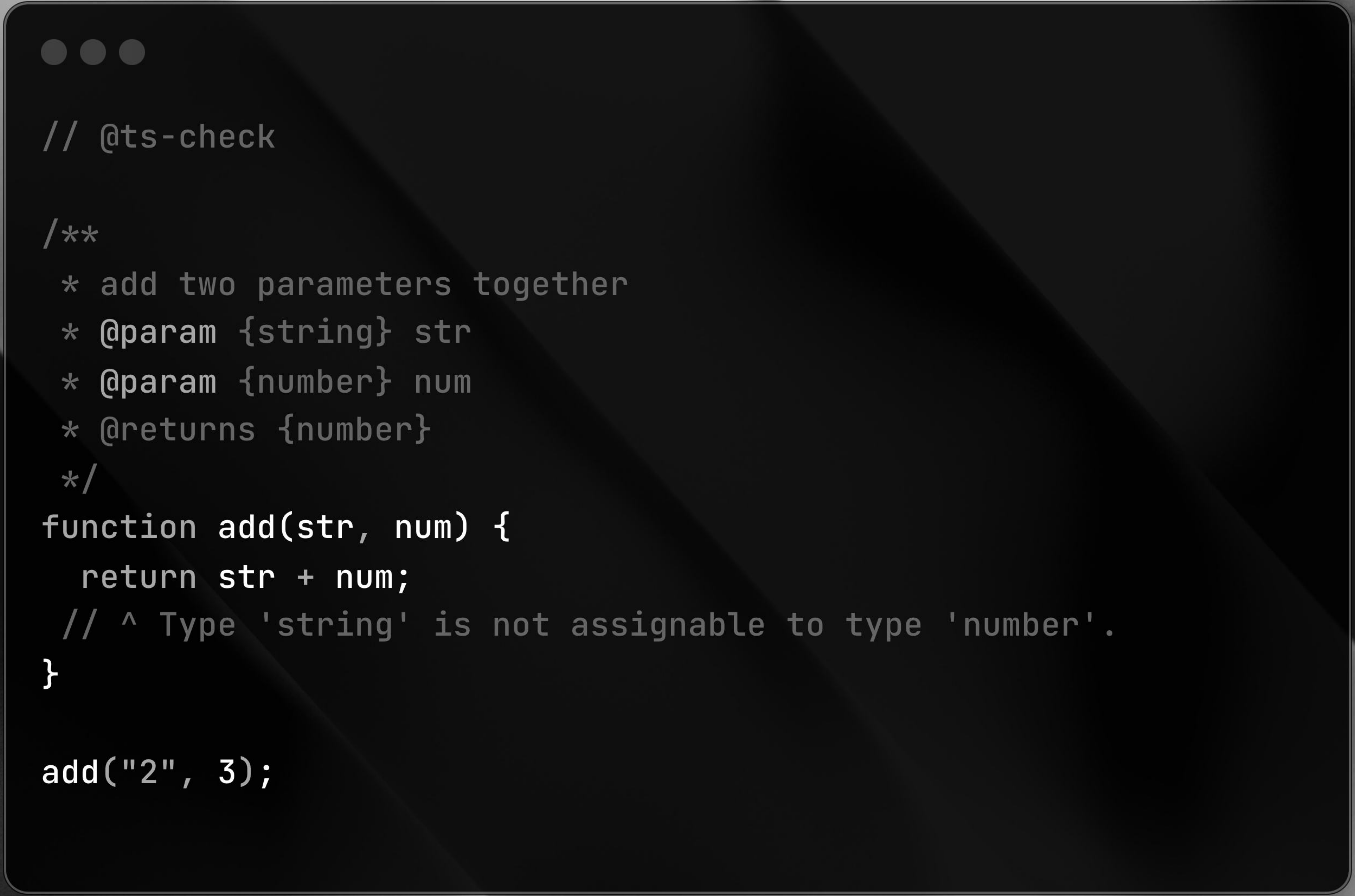


```
/**
 * add two parameters together
 * @param {string} str
 * @param {number} num
 * @returns {number}
 */
function add(str, num) {
  return str + num;
}

add("2", 3);
```

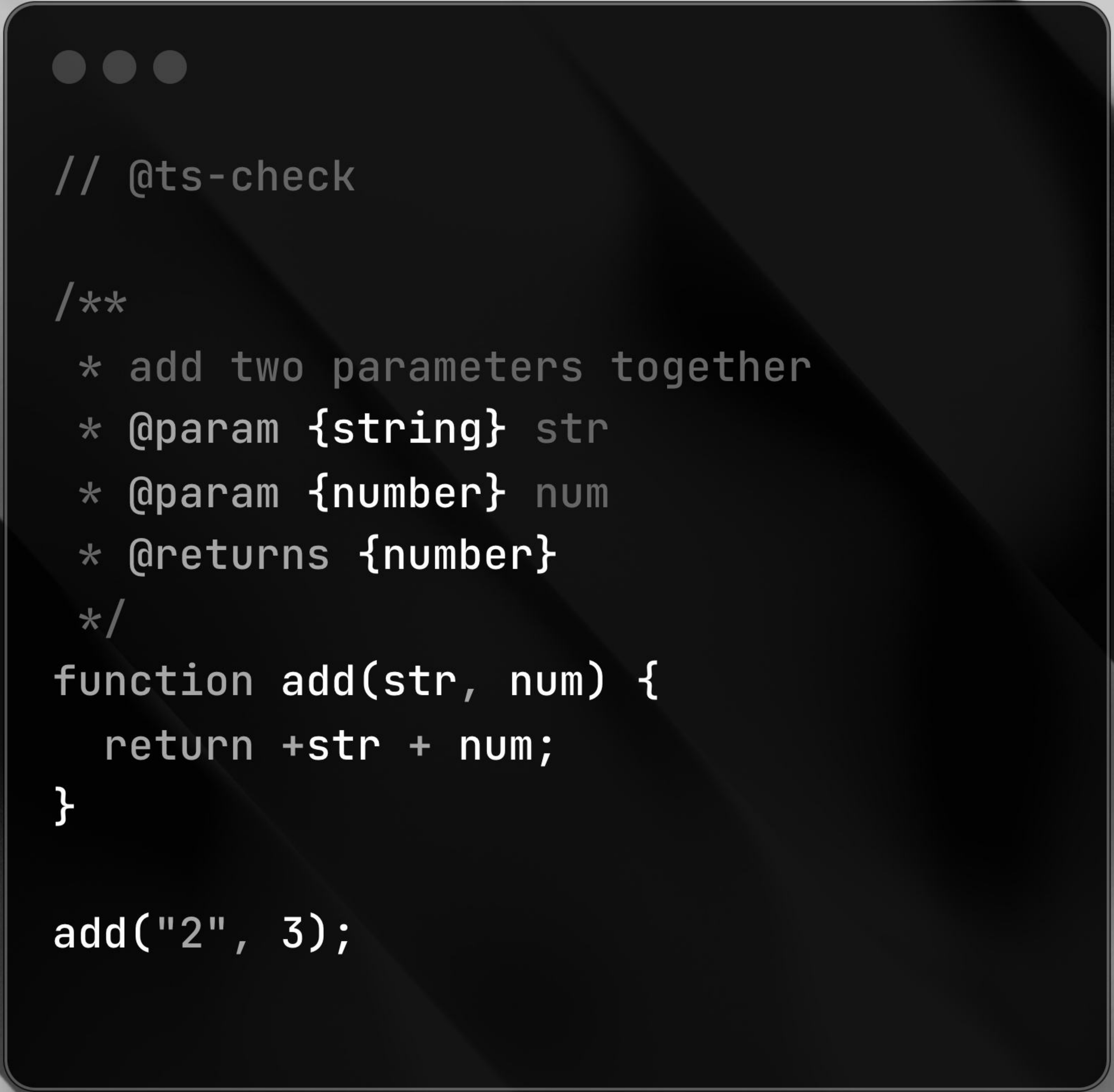


JS 02.jsdoc.2.js > ...

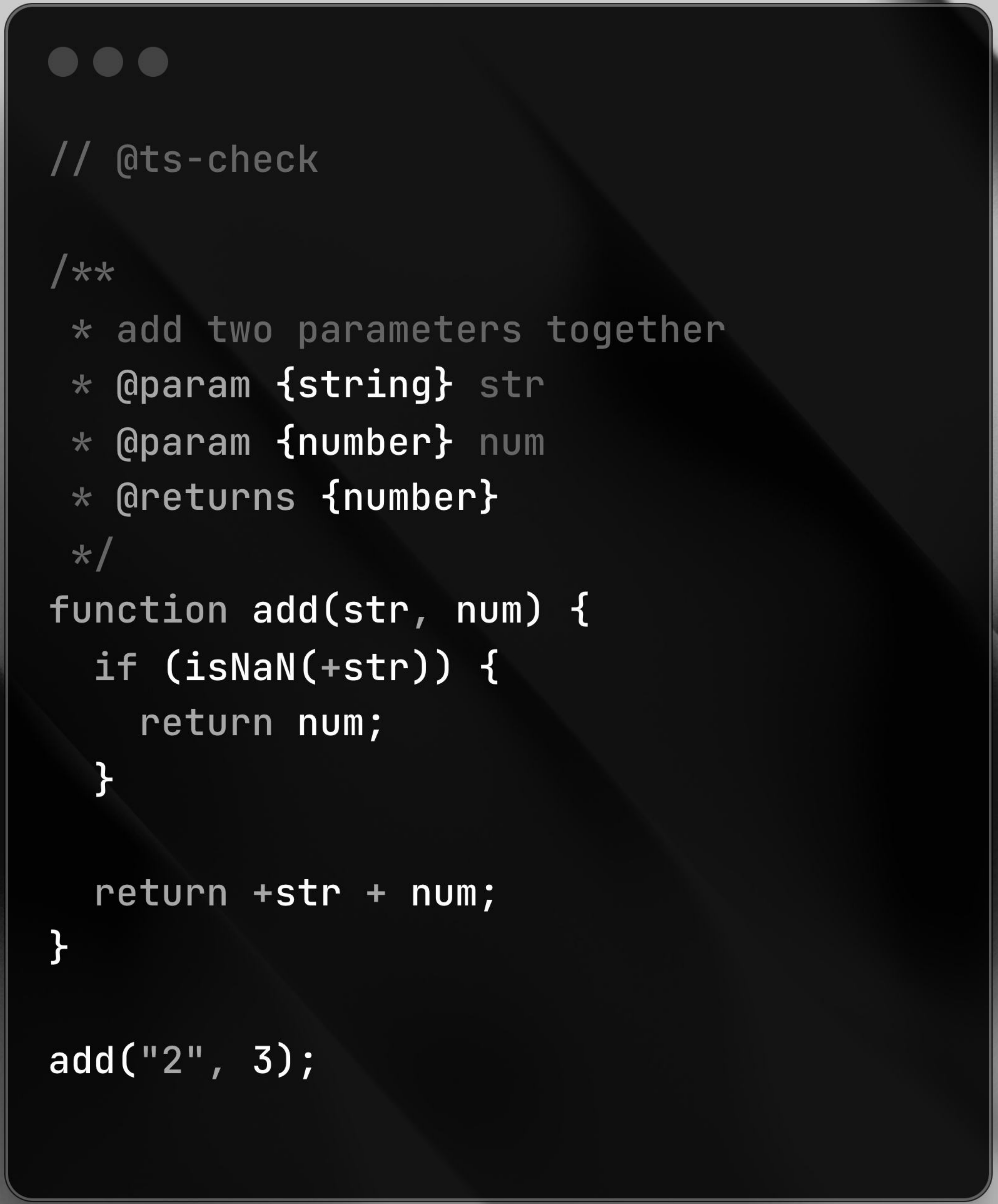
```
// @ts-check

/**
 * add two parameters together
 * @param {string} str
 * @param {number} num
 * @returns {number}
 */
function add(str, num) {
  return str + num;
}

add("2", 3);

```



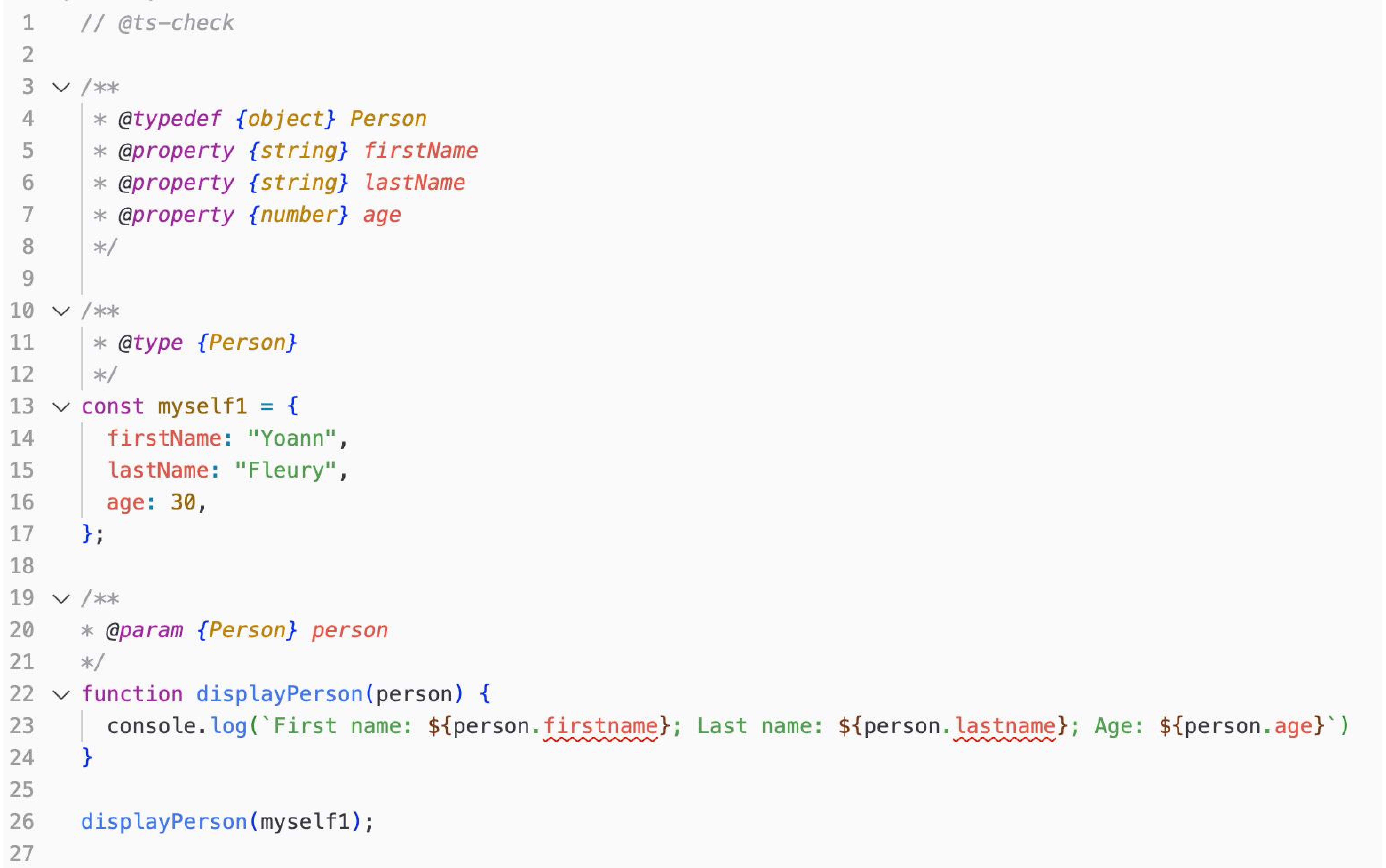
```
// @ts-check

/**
 * add two parameters together
 * @param {string} str
 * @param {number} num
 * @returns {number}
 */
function add(str, num) {
  return str + num;
 // ^ Type 'string' is not assignable to type 'number'.
}

add("2", 3);
```



```
// @ts-check

/**
 * add two parameters together
 * @param {string} str
 * @param {number} num
 * @returns {number}
 */
function add(str, num) {
  return +str + num;
}

add("2", 3);
```



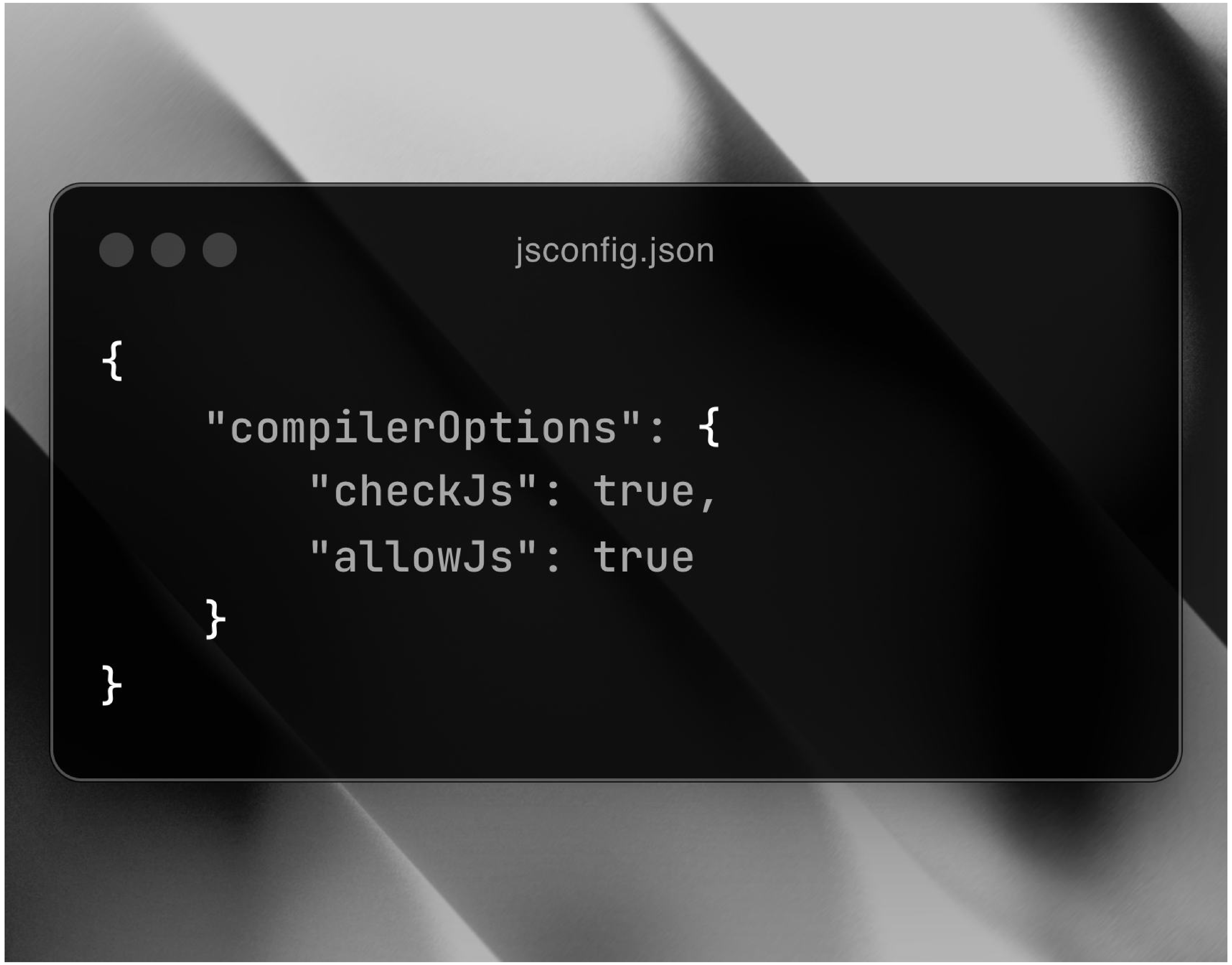
```
// @ts-check

/**
 * add two parameters together
 * @param {string} str
 * @param {number} num
 * @returns {number}
 */
function add(str, num) {
  if (isNaN(+str)) {
    return num;
  }

  return +str + num;
}

add("2", 3);
```



```javascript
// @ts-check

/**
 * @typedef {object} Person
 * @property {string} firstName
 * @property {string} lastName
 * @property {number} age
 */

/**
 * @type {Person}
 */
const myself1 = {
  firstName: "Yoann",
  lastName: "Fleury",
  age: 30,
};

/**
* @param {Person} person
*/
function displayPerson(person) {
  console.log(`First name: ${person.firstname}; Last name: ${person.lastname}; Age: ${person.age}`)
}

displayPerson(myself1);

```

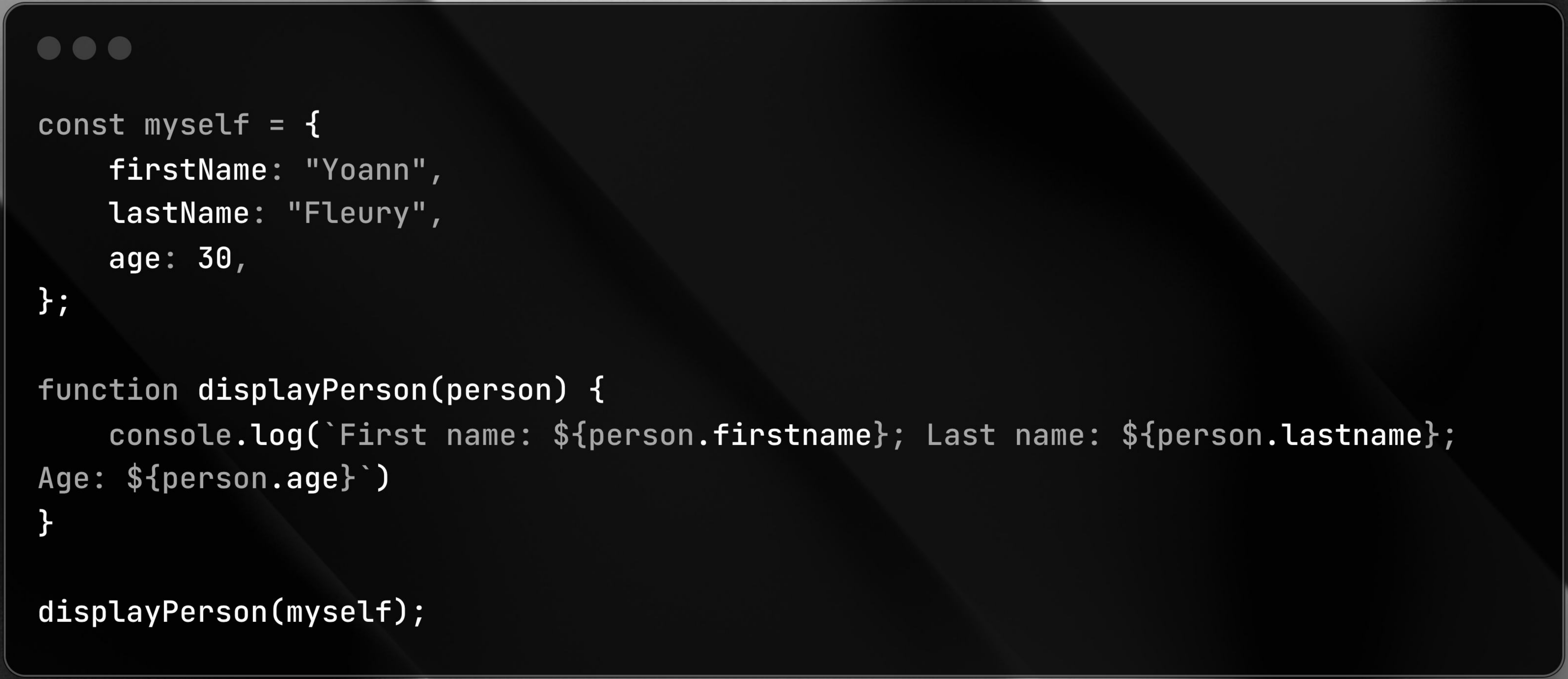
A complete project to check?

Add a **jsconfig.json** file a the root of your project

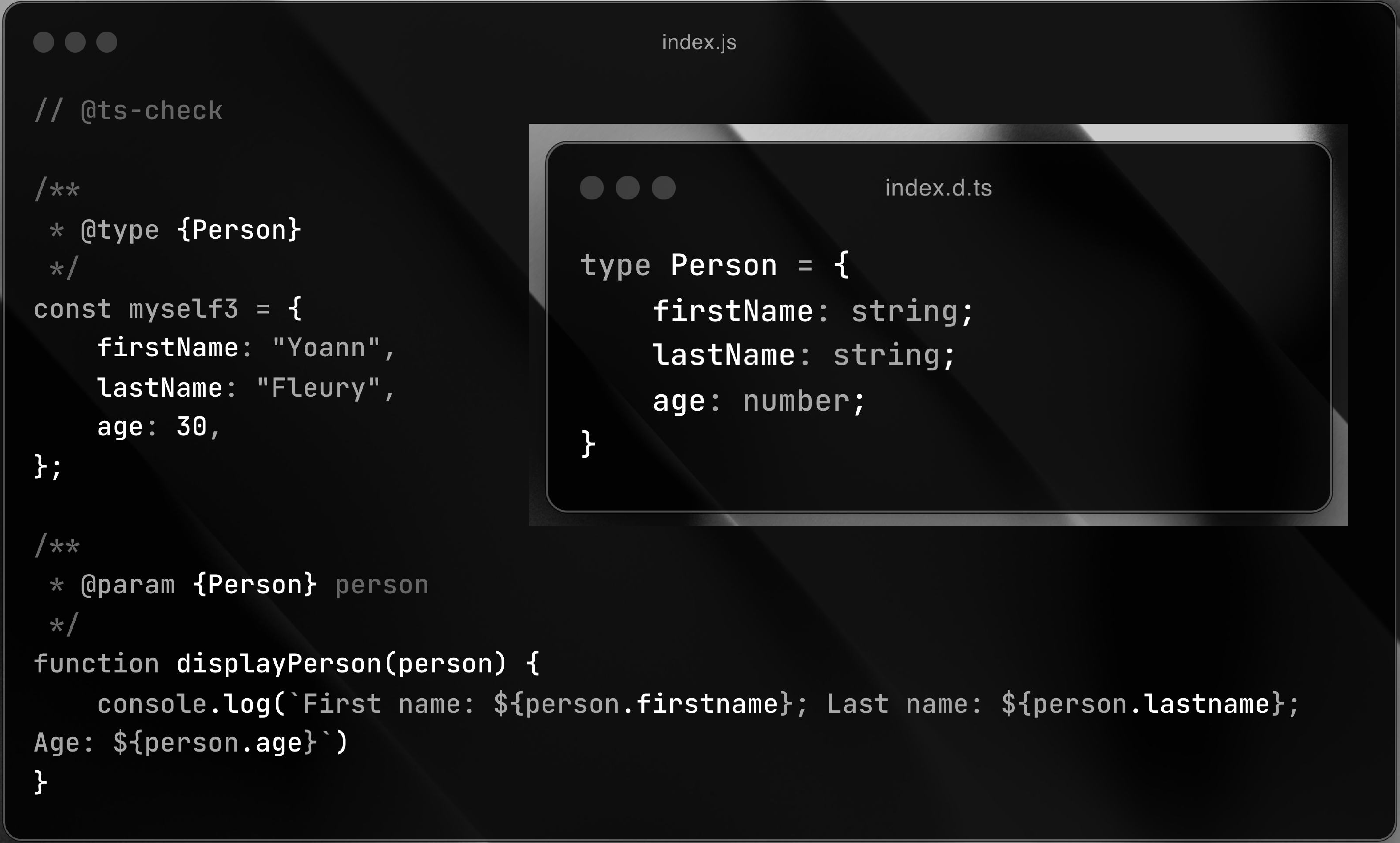


```json
{
    "compilerOptions": {
        "checkJs": true,
        "allowJs": true
    }
}
```

.d.ts

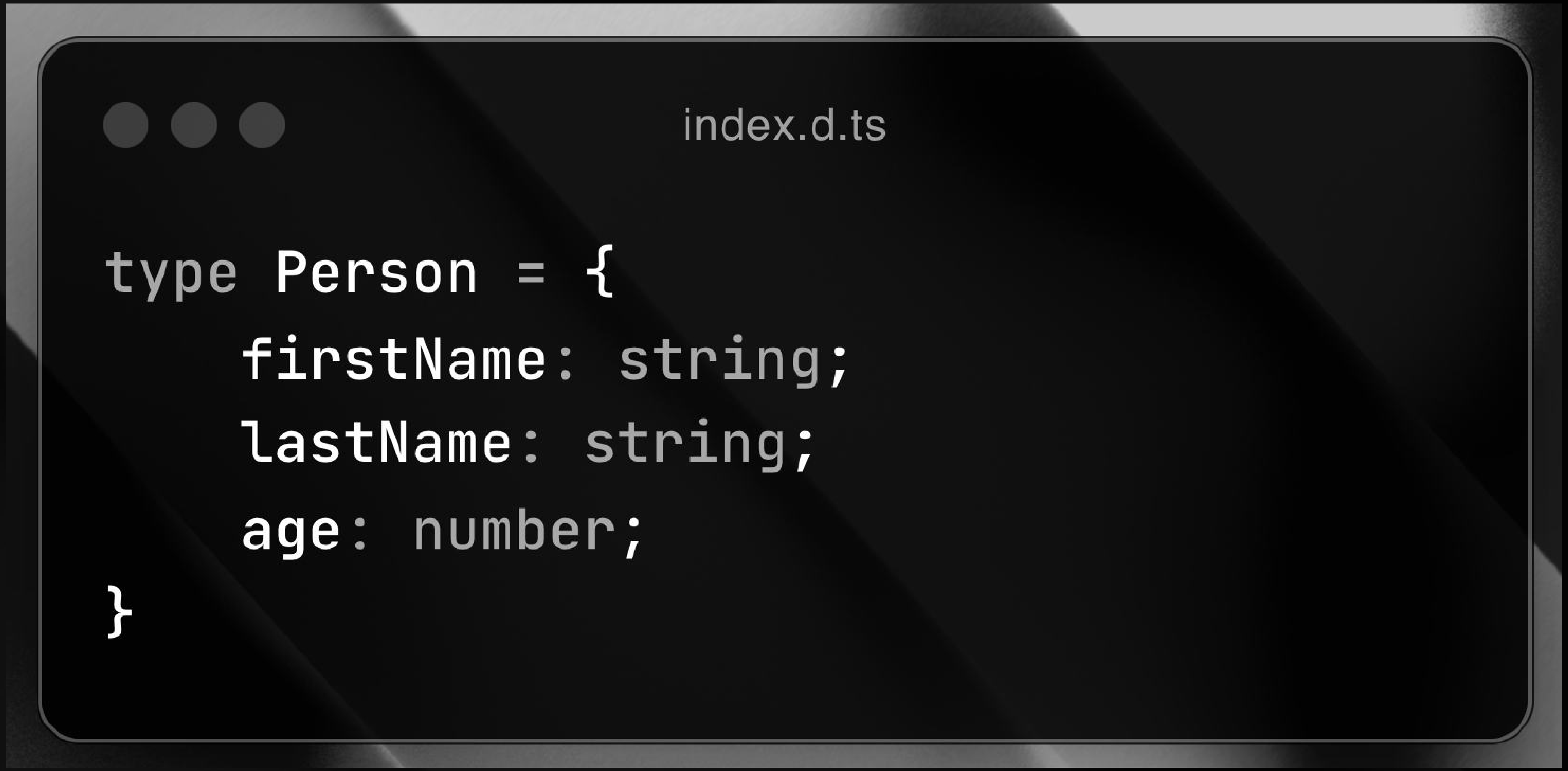


```javascript
const myself = {
    firstName: "Yoann",
    lastName: "Fleury",
    age: 30,
};

function displayPerson(person) {
    console.log(`First name: ${person.firstname}; Last name: ${person.lastname};
Age: ${person.age}`)
}

displayPerson(myself);
```



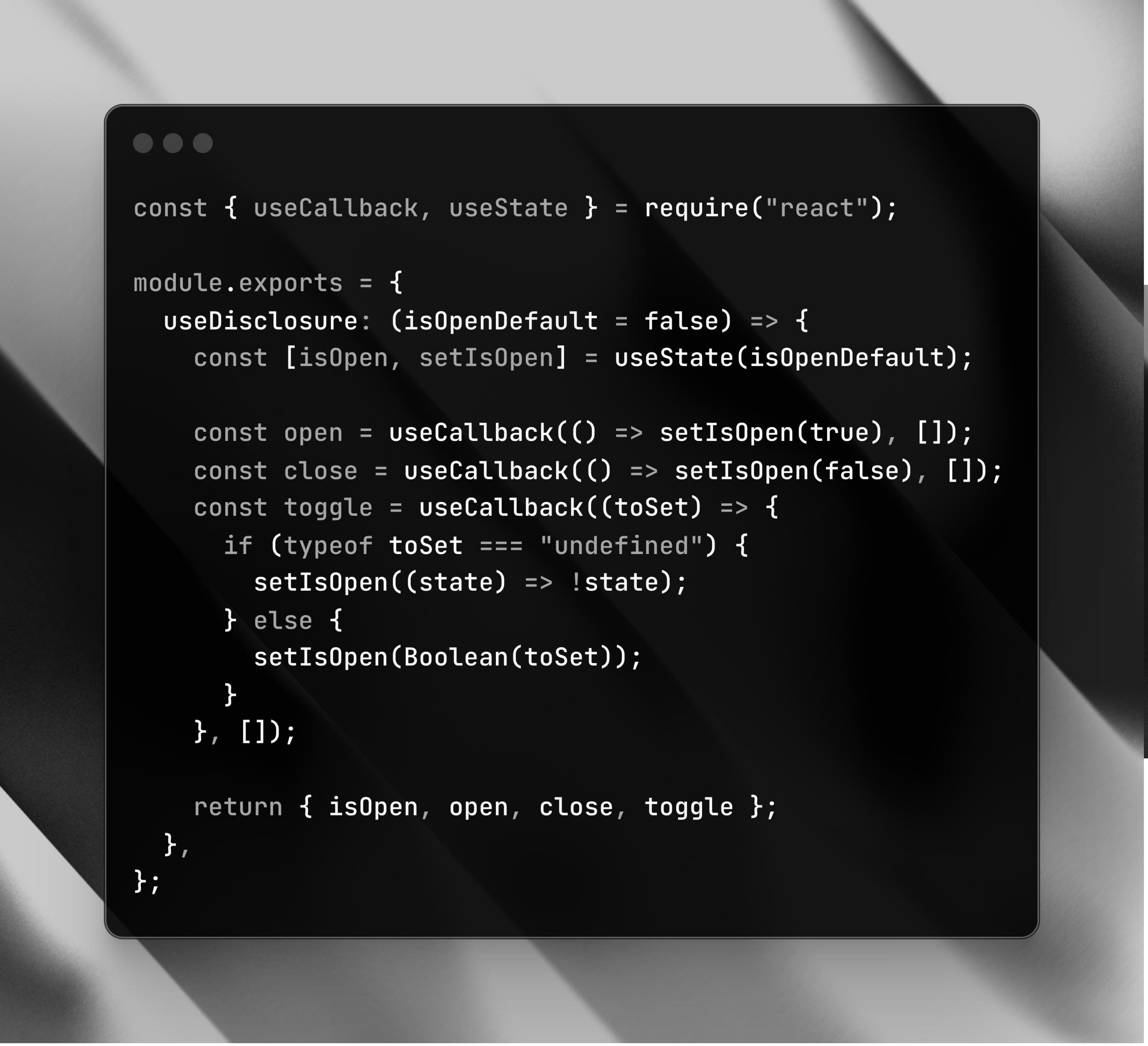
index.js

```js
// @ts-check

/**
 * @type {Person}
 */
const myself3 = {
    firstName: "Yoann",
    lastName: "Fleury",
    age: 30,
};

/**
 * @param {Person} person
 */
function displayPerson(person) {
    console.log(`First name: ${person.firstname}; Last name: ${person.lastname};
Age: ${person.age}`)
}
```

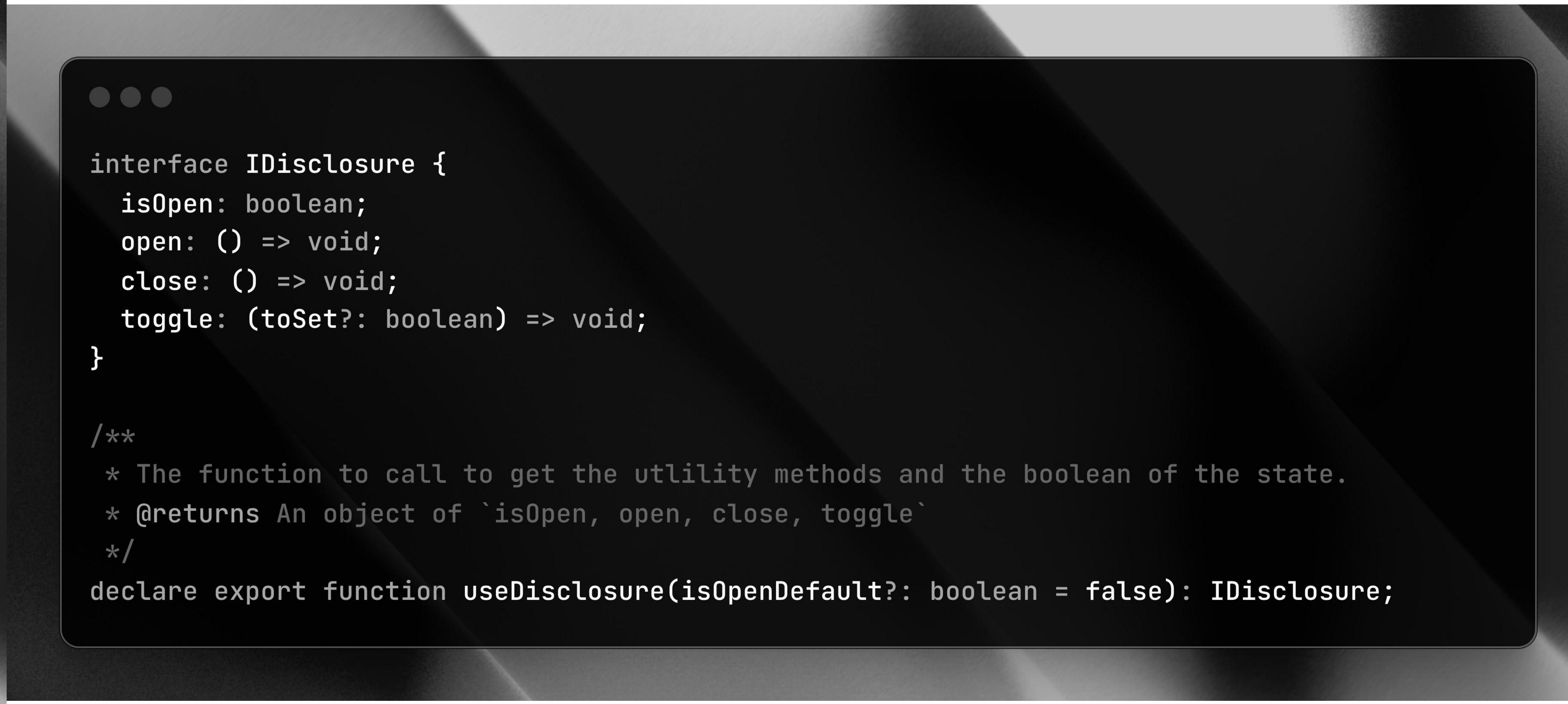


index.d.ts

```ts
type Person = {
    firstName: string;
    lastName: string;
    age: number;
}
```



```js
const { useCallback, useState } = require("react");

module.exports = {
  useDisclosure: (isOpenDefault = false) => {
    const [isOpen, setIsOpen] = useState(isOpenDefault);

    const open = useCallback(() => setIsOpen(true), []);
    const close = useCallback(() => setIsOpen(false), []);
    const toggle = useCallback((toSet) => {
      if (typeof toSet === "undefined") {
        setIsOpen((state) => !state);
      } else {
        setIsOpen(Boolean(toSet));
      }
    }, []);

    return { isOpen, open, close, toggle };
  },
};
```

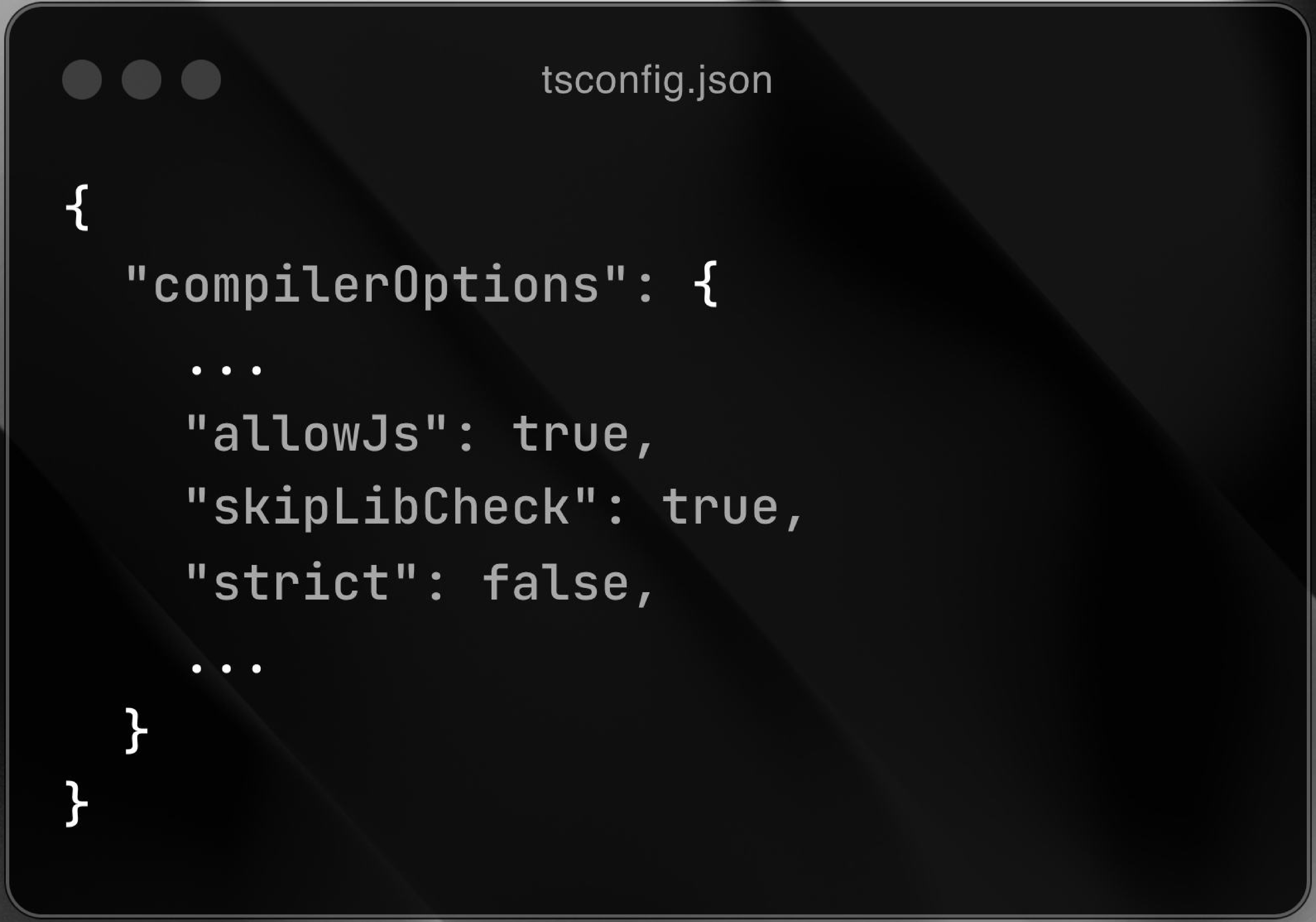


```ts
interface IDisclosure {
  isOpen: boolean;
  open: () => void;
  close: () => void;
  toggle: (toSet?: boolean) => void;
}

/**
 * The function to call to get the utlility methods and the boolean of the state.
 * @returns An object of `isOpen, open, close, toggle`
 */
declare export function useDisclosure(isOpenDefault?: boolean = false): IDisclosure;
```

```
npm i react-use-disclosure
```

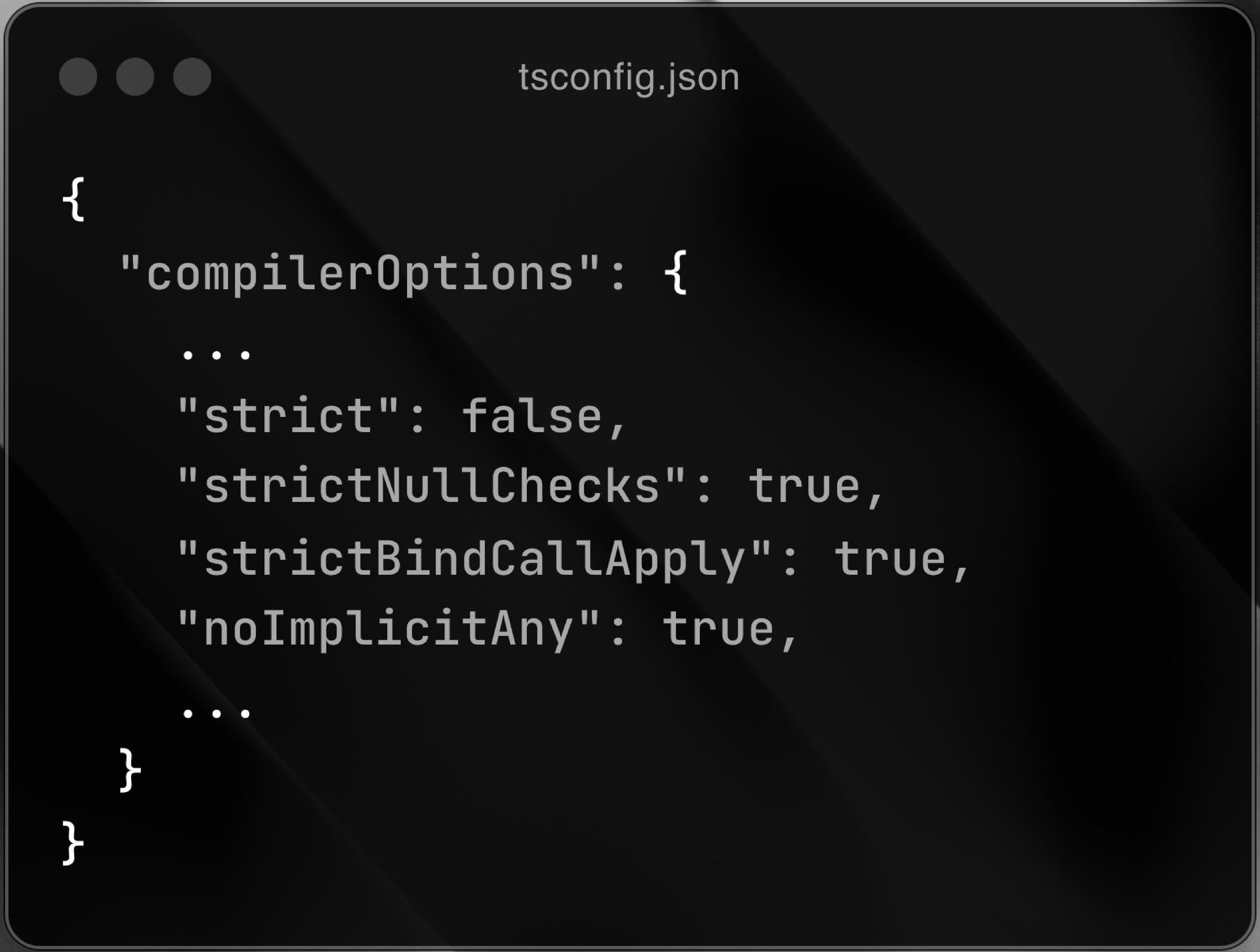
Write TypeScript (for the type system)

tsconfig


tsconfig.json

```json
{
  "compilerOptions": {
    ...
    "allowJs": true,
    "skipLibCheck": true,
    "strict": false,
    ...
  }
}
```

strictness


tsconfig.json

```
{
  "compilerOptions": {
    ...
    "strict": false,
    "strictNullChecks": true,
    "strictBindCallApply": true,
    "noImplicitAny": true,
    ...
  }
}
```

@types/

react-native-ficus-ui TS

1.1.0 • Public • Published 12

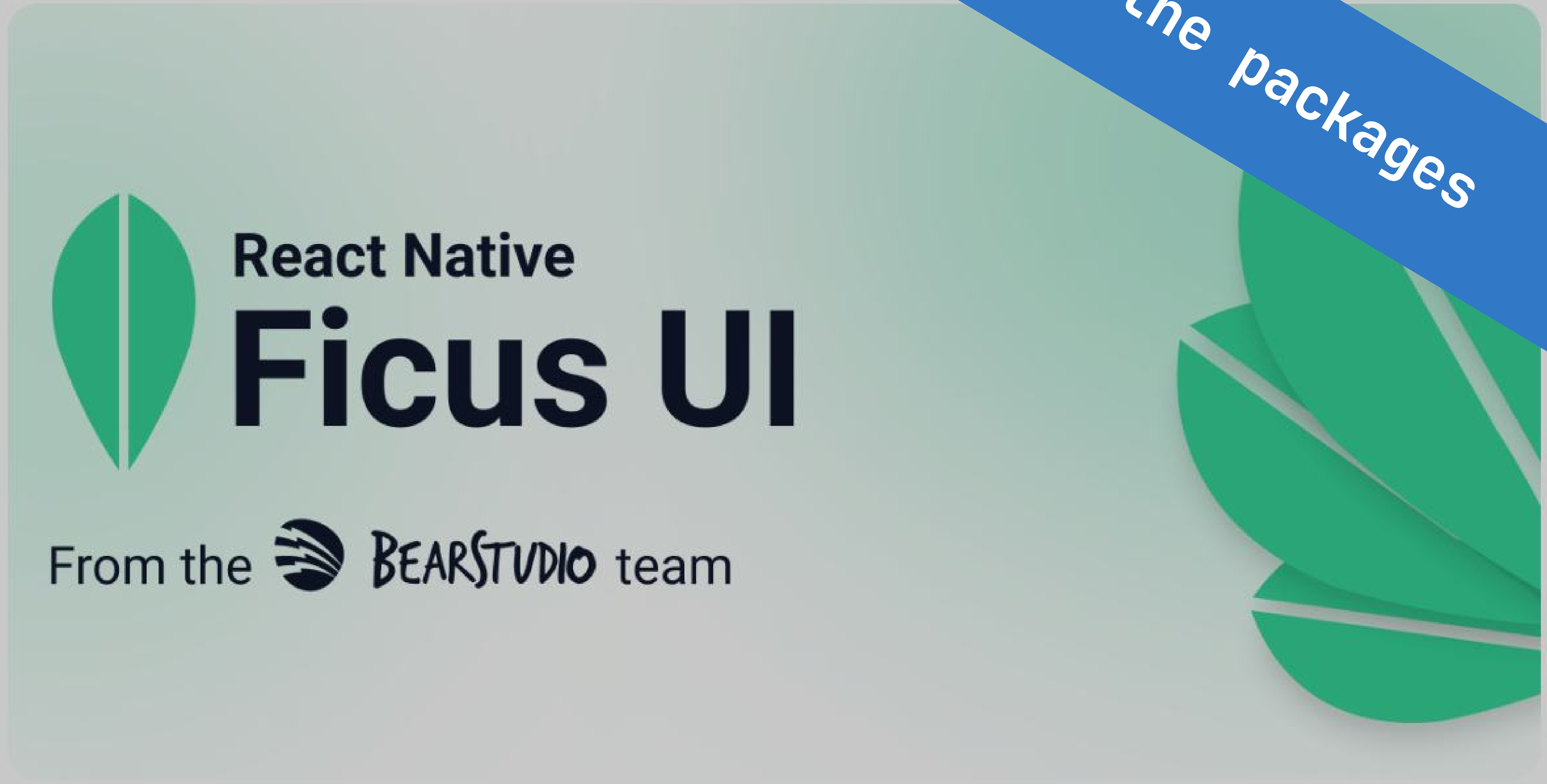
Types in the packages

Readme

9 Dependencies

0 Dependents

27 Versions



Install

```
> npm i react-native-ficus-ui
```

Repository

github.com/BearStudio/react-native-fic...

Homepage

ficus-ui.com

Weekly Downloads

27

Ficus UI is a React Native UI library forked on Magnus UI and inspired by Chakra UI

Version

License

MIT

Installation

With pnpm :

```
npm i react-native-ficus-ui
```

ed Size

1.56 MB

Total Files

962

bcrypt DT

5.1.1 • Publi d a year ago

Readr de Beta | 2 Dependencies | 7,127 Dependents | 54 Versions

Types provided by Definitely Typed

node.bcrypt.js

ci passing

A library to help you hash passwords.

You can read about bcrypt in Wikipedia as well as in the following article: How To Safely Store A Password

If You Are Submitting Bugs or Issues

Please verify that the NodeJS version you are using is a *stable* version; Unstable versions are currently not supported and issues created whi

If you are on a stable version of NodeJS, please installation issues. The code snippet does not r However, it must provide enough information so the problem can be replicable, or it may be

```
npm i bcrypt
npm i --include=dev @types/bcrypt
```

Install

```
> npm i bcrypt
```

Repository

github.com/kelektiv/node.bcrypt.js

Homepage

github.com/kelektiv/node.bcrypt.js#rea...

Weekly Downloads

1,758,677

Version	License
	MIT

ked Size	Total Files
111 kB	26

create-start-ui

0.6.0 • Public • Publi r ago

Readme | 13 Dependencies | 0 Dependents | 9 Versions | Settings

Create a 🚀 Start UI pr

Usage

Generate a 🚀 Start UI project in a new folder.

```
yarn create start-ui --web [projectName]    # Generate a start-ui-web proj
yarn create start-ui --native [projectName] # Generate a start-ui-native p
```

Options

```
-h, --help                 Show this help
-v, --version              Display CLI
--web PROJECT_PATH         Scaffold a
--native PROJECT_PATH      Scaffold a
--branch BRANCH_NAME       Specify the
--no-git-init              Ignore `git init` step
```

```
npx create-start-ui
```

Install

```
> npm i create-start-ui
```

Repository

github.com/bearstudio/create-start-ui

Homepage

github.com/bearstudio/create-start-ui#r...

Weekly Downloads

11

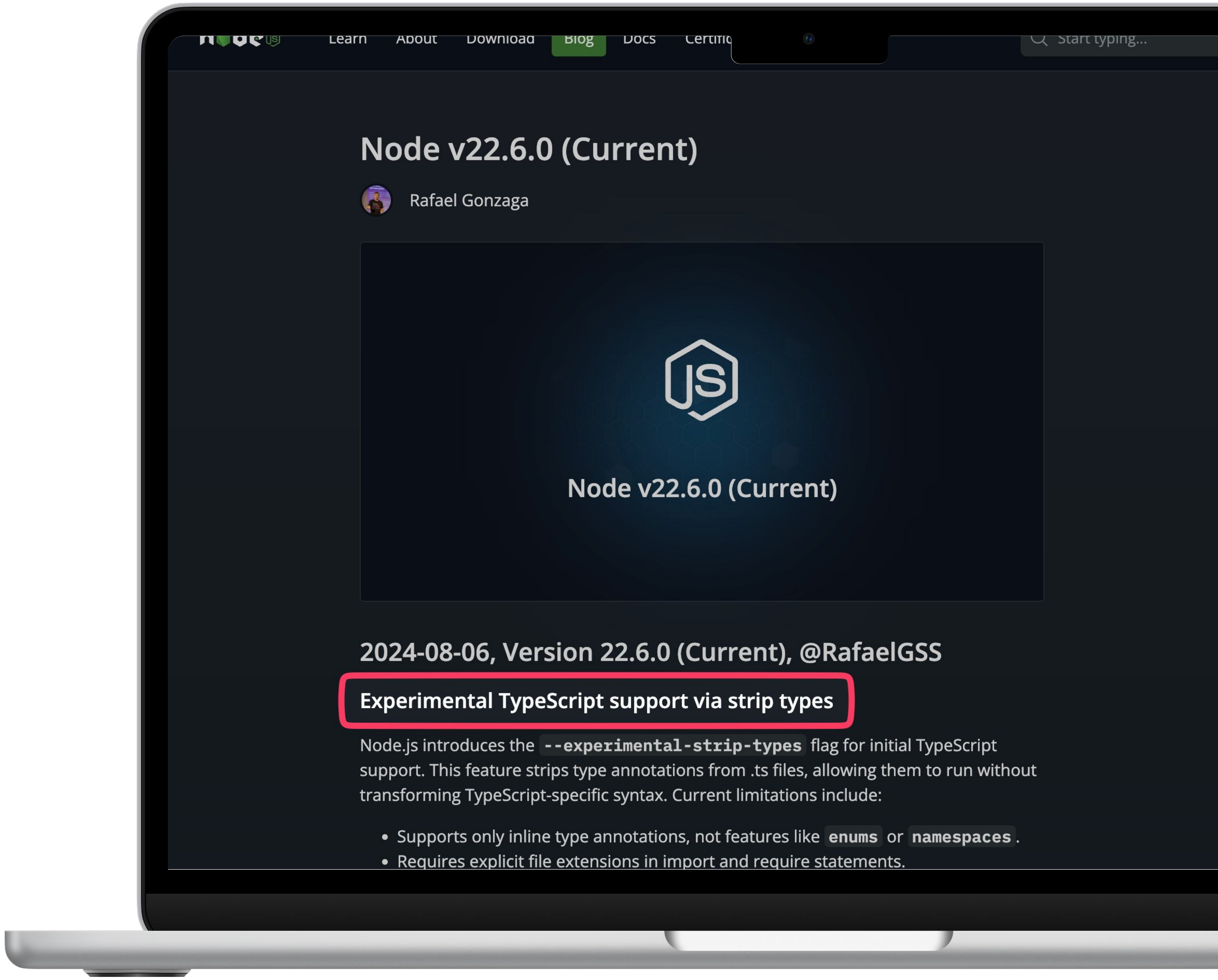
Version	License
	MIT

ked Size	Total Files
25.1 kB	16

Don’t use fancy features

Experimental TypeScript support in Node.js

--experimental-strip-types flag to run TypeScript code directly from Node, no more compilation time!



Node v22.6.0 (Current)

Rafael Gonzaga

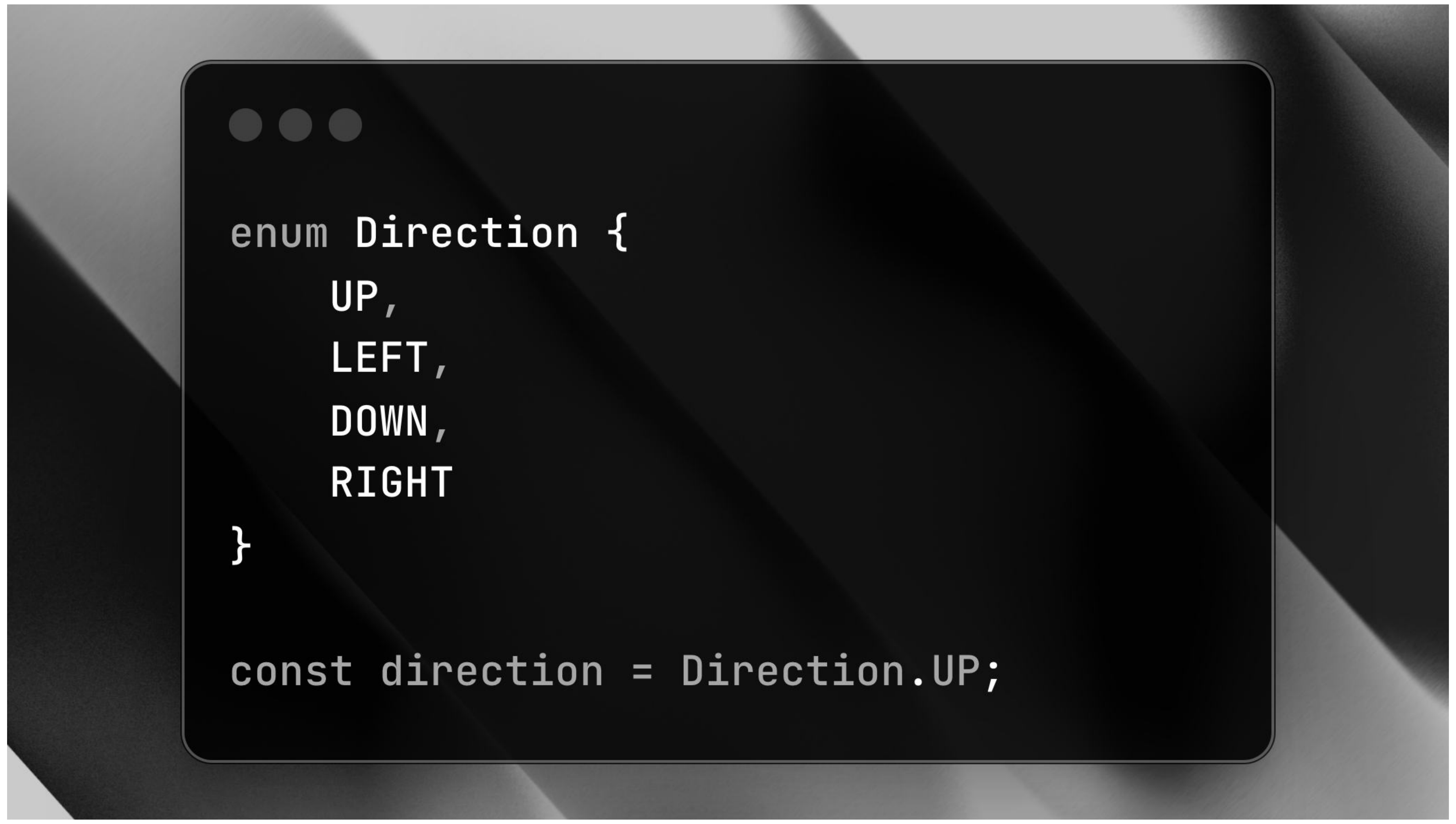
2024-08-06, Version 22.6.0 (Current), @RafaelGSS

Experimental TypeScript support via strip types

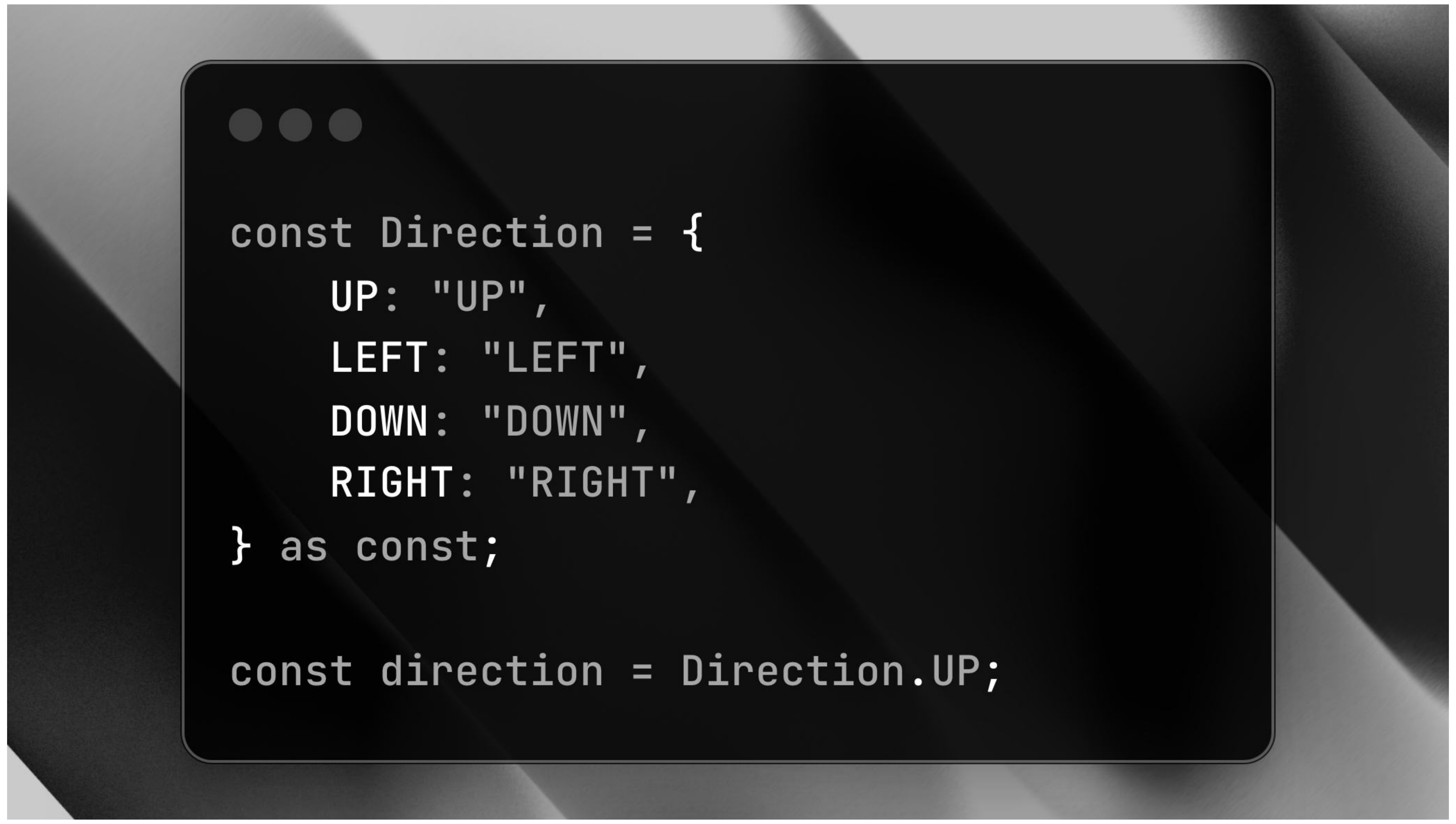
Node.js introduces the `--experimental-strip-types` flag for initial TypeScript support. This feature strips type annotations from .ts files, allowing them to run without transforming TypeScript-specific syntax. Current limitations include:

- Supports only inline type annotations, not features like `enums` or `namespaces`.
- Requires explicit file extensions in import and require statements.

Don’t use: enums



```
enum Direction {
    UP,
    LEFT,
    DOWN,
    RIGHT
}

const direction = Direction.UP;
```



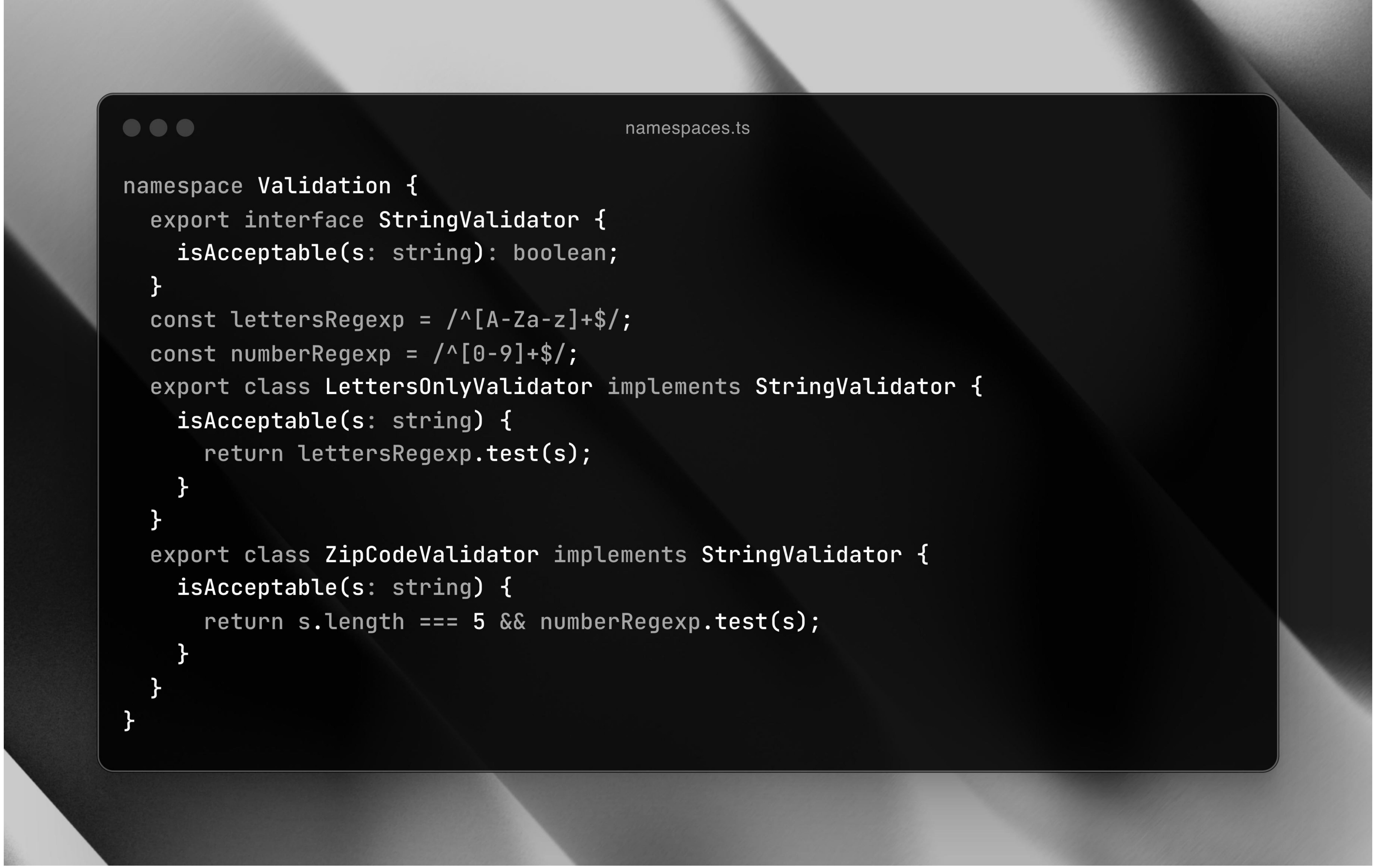
```
const Direction = {
    UP: "UP",
    LEFT: "LEFT",
    DOWN: "DOWN",
    RIGHT: "RIGHT",
} as const;

const direction = Direction.UP;
```

Don’t

You can’t easily use the enum, it is not an enum like in other languages

Do

Use object as const


namespaces.ts

```ts
namespace Validation {
  export interface StringValidator {
    isAcceptable(s: string): boolean;
  }
  const lettersRegexp = /^[A-Za-z]+$/;
  const numberRegexp = /^[0-9]+$/;
  export class LettersOnlyValidator implements StringValidator {
    isAcceptable(s: string) {
      return lettersRegexp.test(s);
    }
  }
  export class ZipCodeValidator implements StringValidator {
    isAcceptable(s: string) {
      return s.length === 5 && numberRegexp.test(s);
    }
  }
}
```

Don’t use: namespace

You just don’t need them

Don’t use

Fancy TypeScript features.
They are only syntactic sugar.

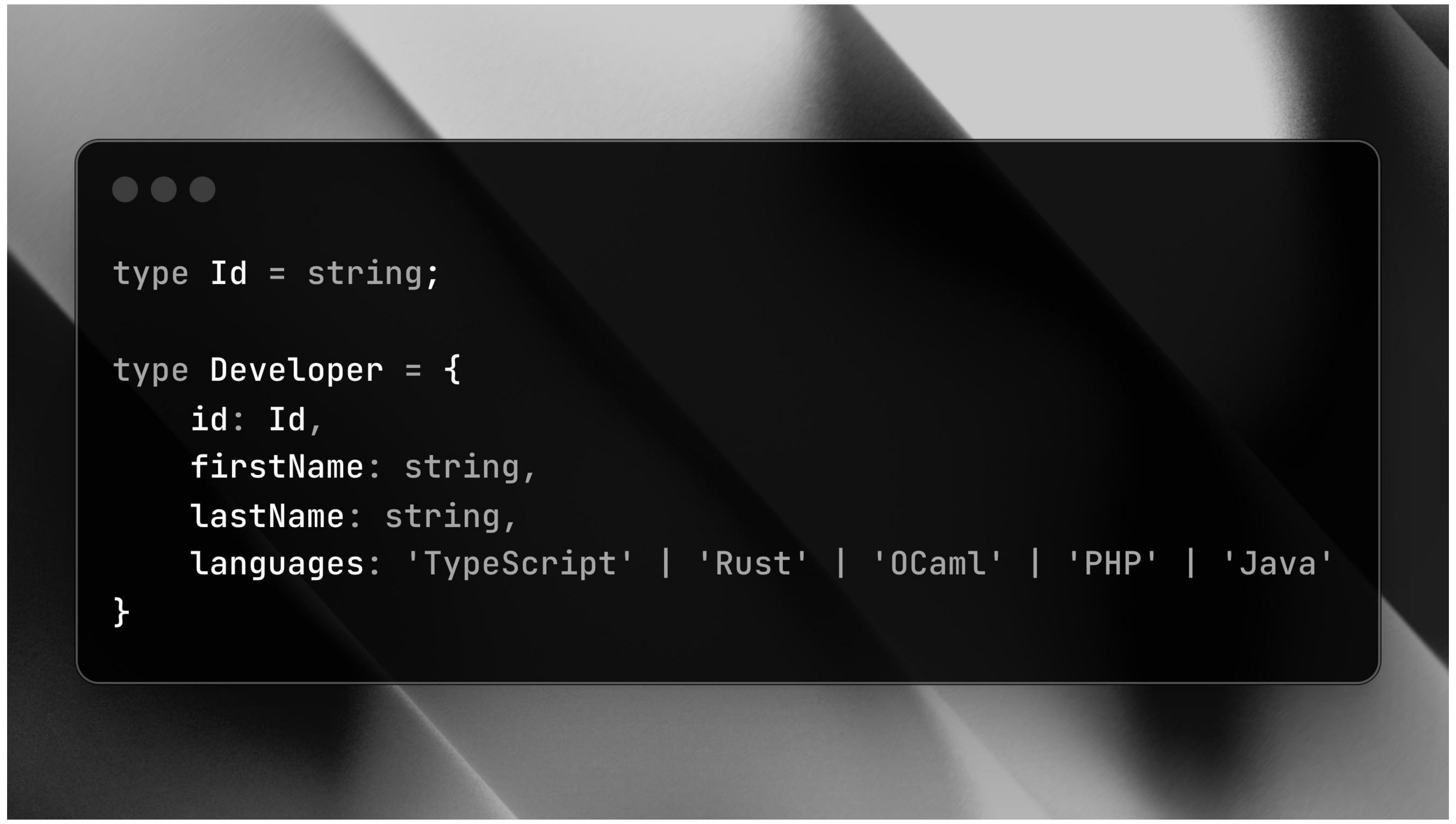
- ❌ Decorators: `@something()`
- ❌ enums
- ❌ namespaces

Use the type system

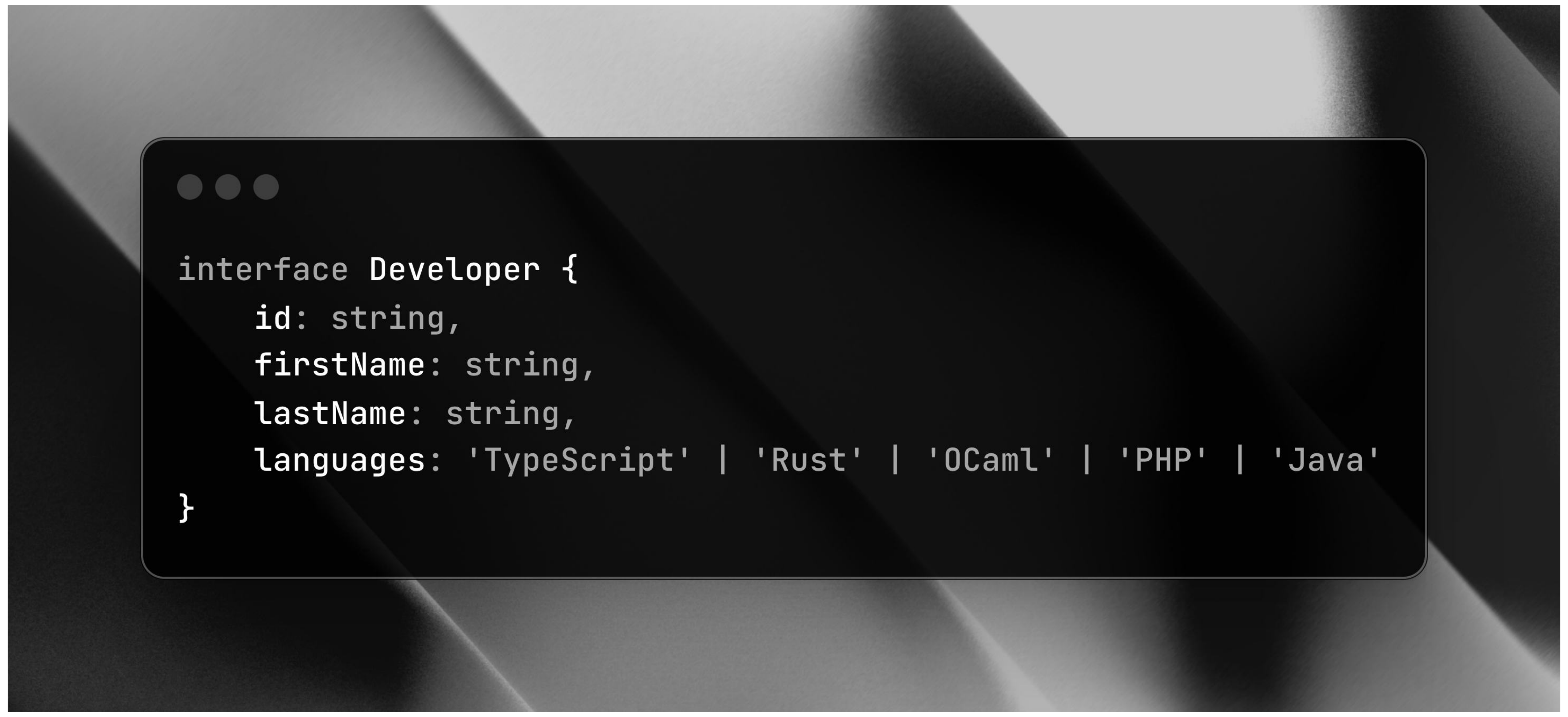
Prefer Type

over Interface

Prefer Type



```
type Id = string;

type Developer = {
    id: Id,
    firstName: string,
    lastName: string,
    languages: 'TypeScript' | 'Rust' | 'OCaml' | 'PHP' | 'Java'
}
```



```
interface Developer {
    id: string,
    firstName: string,
    lastName: string,
    languages: 'TypeScript' | 'Rust' | 'OCaml' | 'PHP' | 'Java'
}
```

Types

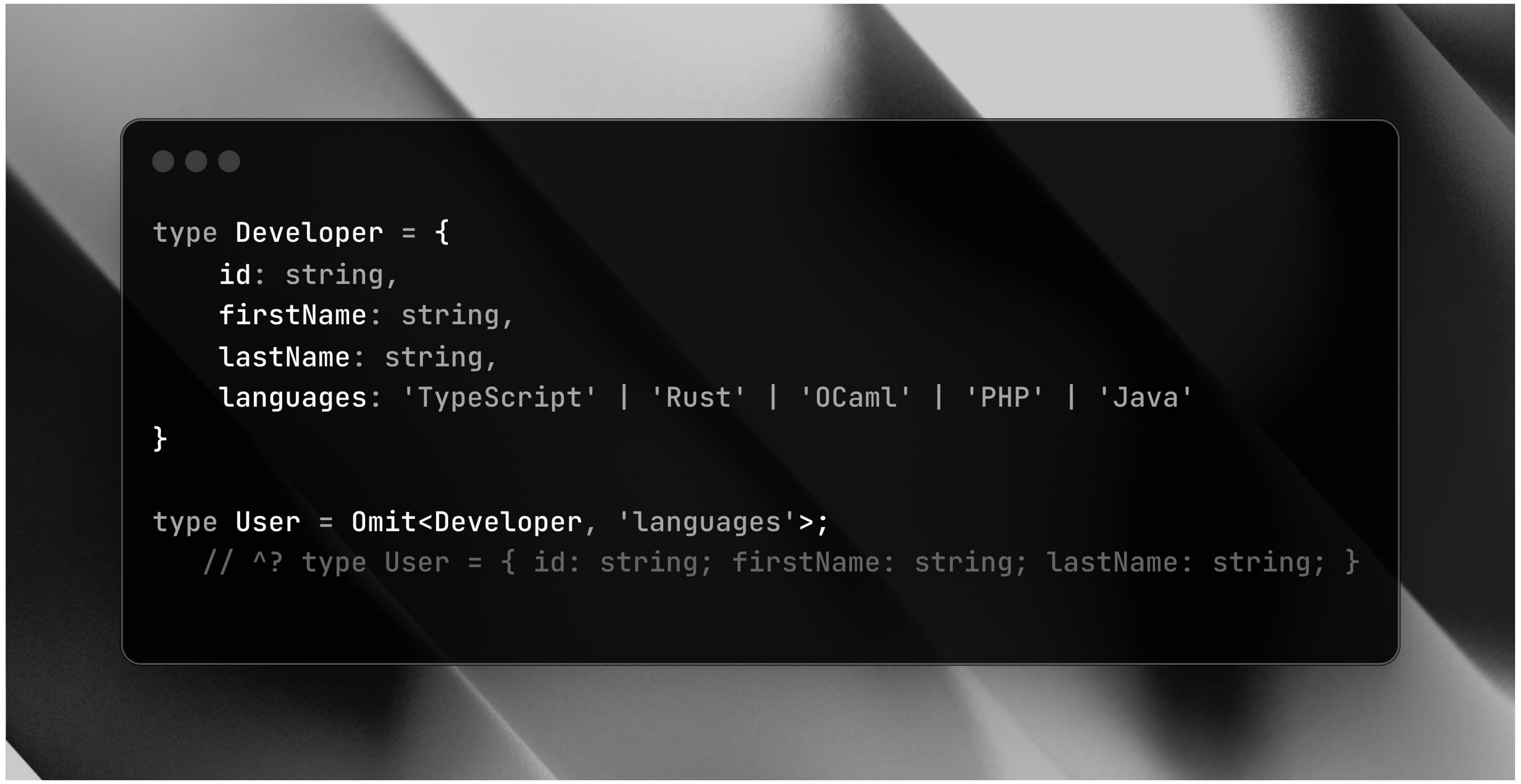
- Create alias
- Concise type declaration

Interface

- OOP oriented
- Interfaces will merge their attributes

Utils

Pick<T>, Omit<T>, Required<T>,
Partials<T>, Awaited<T>, ...



```
type Developer = {
    id: string,
    firstName: string,
    lastName: string,
    languages: 'TypeScript' | 'Rust' | 'OCaml' | 'PHP' | 'Java'
}

type User = Omit<Developer, 'languages'>;
   // ^? type User = { id: string; firstName: string; lastName: string; }
```

TypeScript first libraries

zod

runtime validator and static type generator

remeda

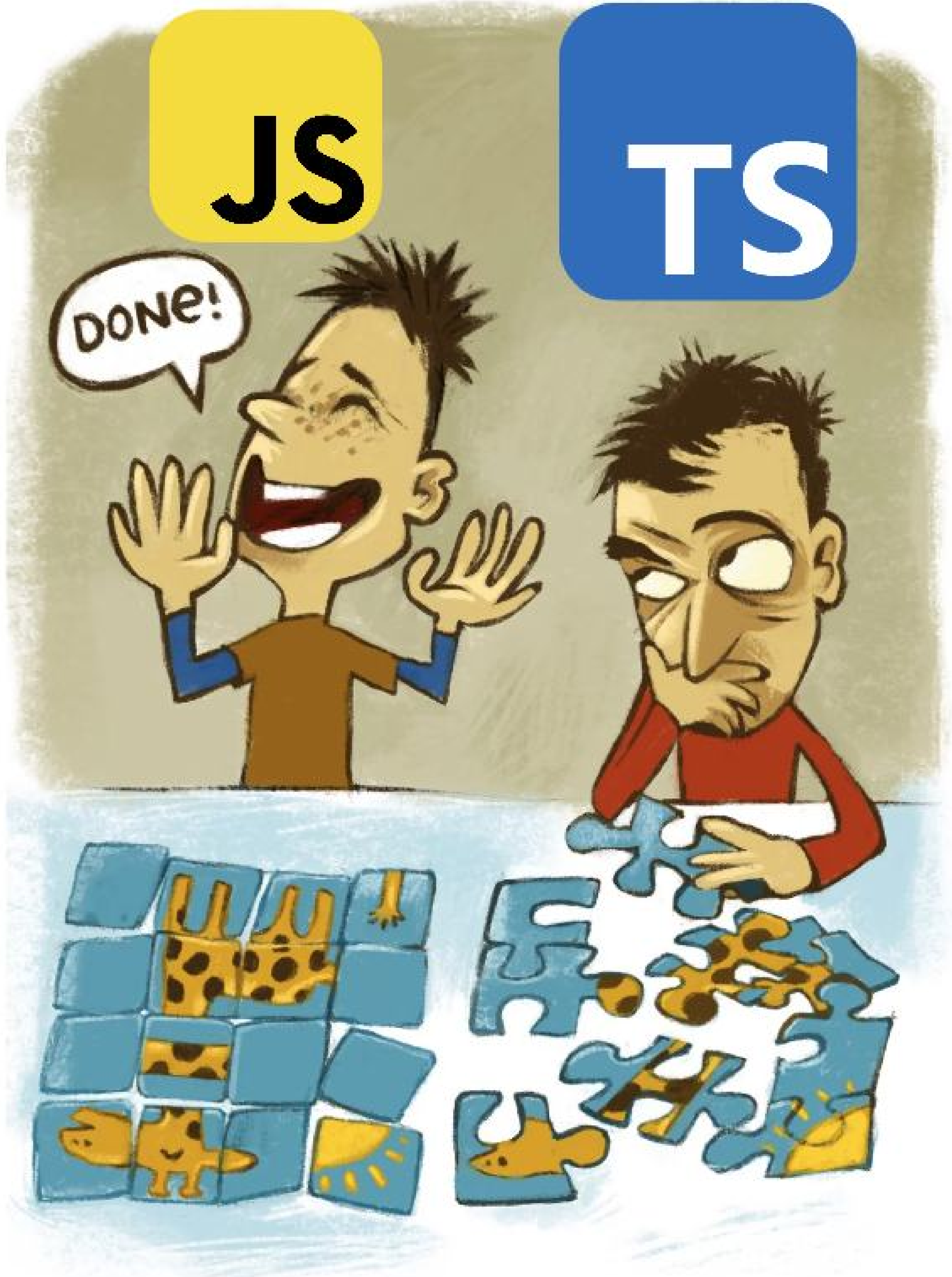
a set of utils for better type inference (like lodash but better)

ts-pattern

an improved switch/case for TypeScript

🇫🇷 yoannfleury.dev/blog/mes-libs-typescript-du-moment/

Conclusion


JS
TS
DONE!

TYPES

TYPES

TYPES

Questions?