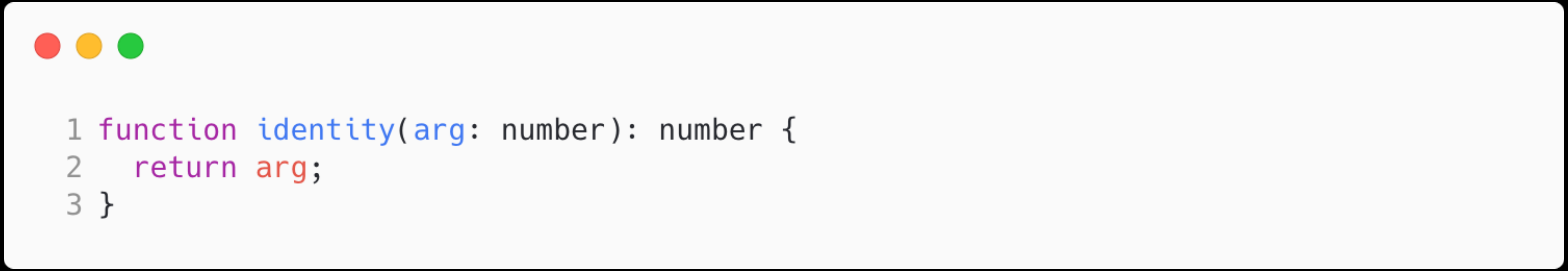
进阶 TypeScript

初见泛型

https://github.com/zongzi531

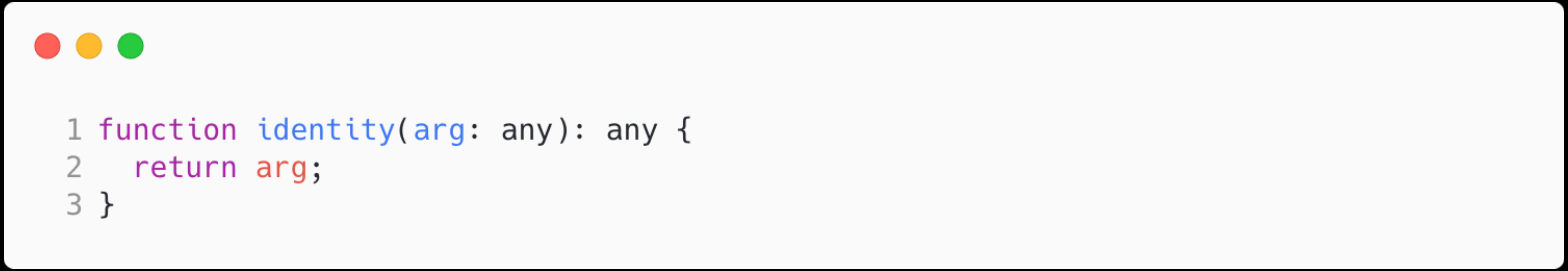
为什么会有泛型

为什么会有泛型



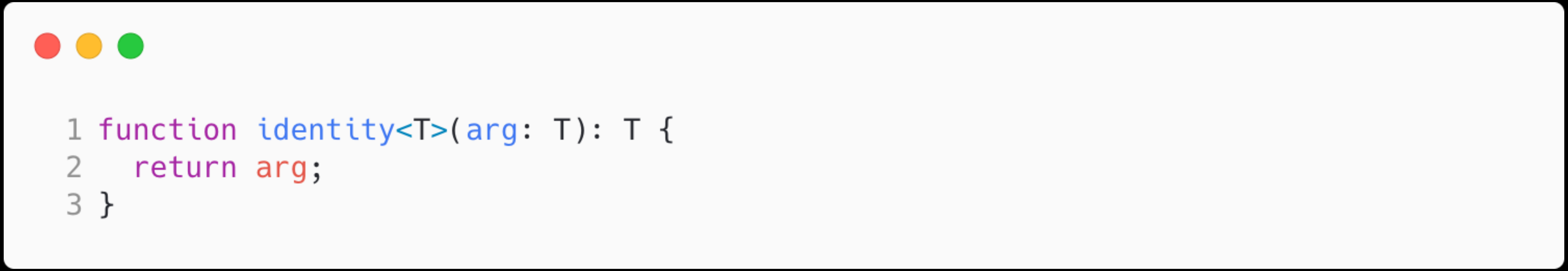
```
function identity(arg: number): number {
  return arg;
}
```

为什么会有泛型



```
function identity(arg: any): any {
  return arg;
}
```

为什么会有泛型



```
function identity<T>(arg: T): T {
  return arg;
}
```

Partial<Type>

ThisType<Type>

Extract<Type, Union>

Exclude<Type, ExcludedUnion>

Required<Type>

Readonly<Type>

NonNullable<Type>

ThisParameterType<Type>

Parameters<Type>

InstanceType<Type>

官方提供的全局泛型

Pick<Type, Keys>

Lowercase<StringType>

Omit<Type, Keys>

ReturnType<Type>

ConstructorParameters<Type>

Record<Keys,Type>

Uncapitalize<StringType>

OmitThisParameter<Type>

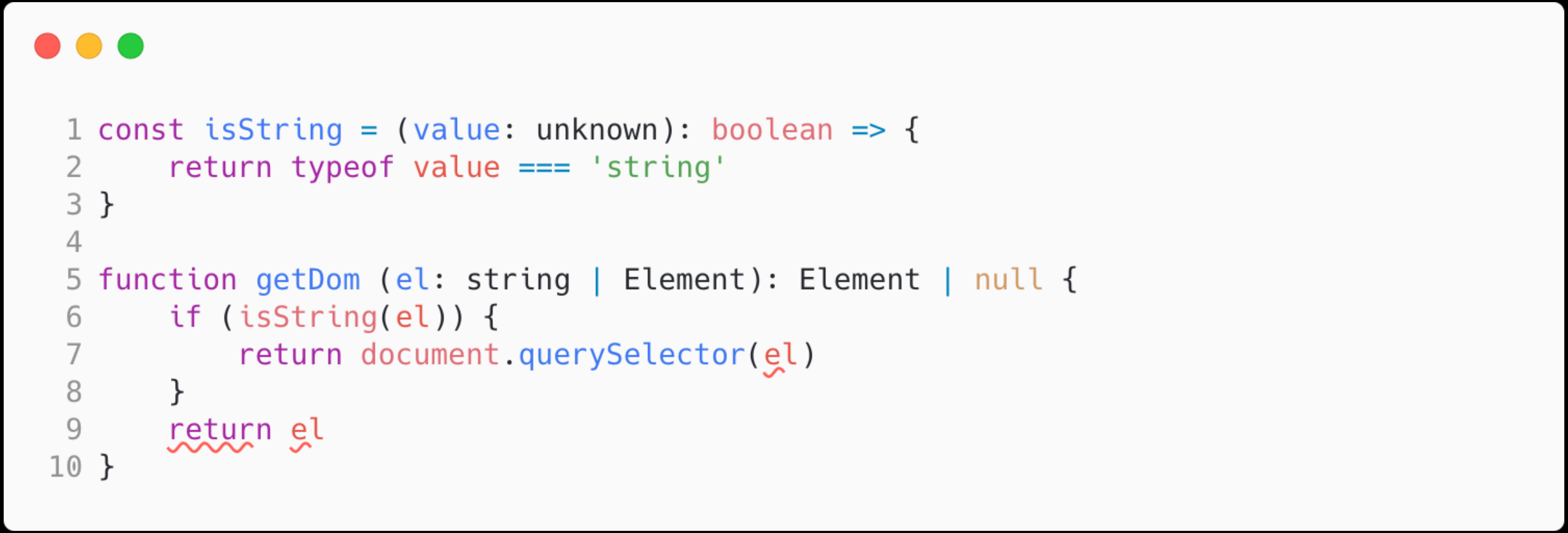
Uppercase<StringType>

Capitalize<StringType>

如何自己书写泛型

如何自己书写泛型

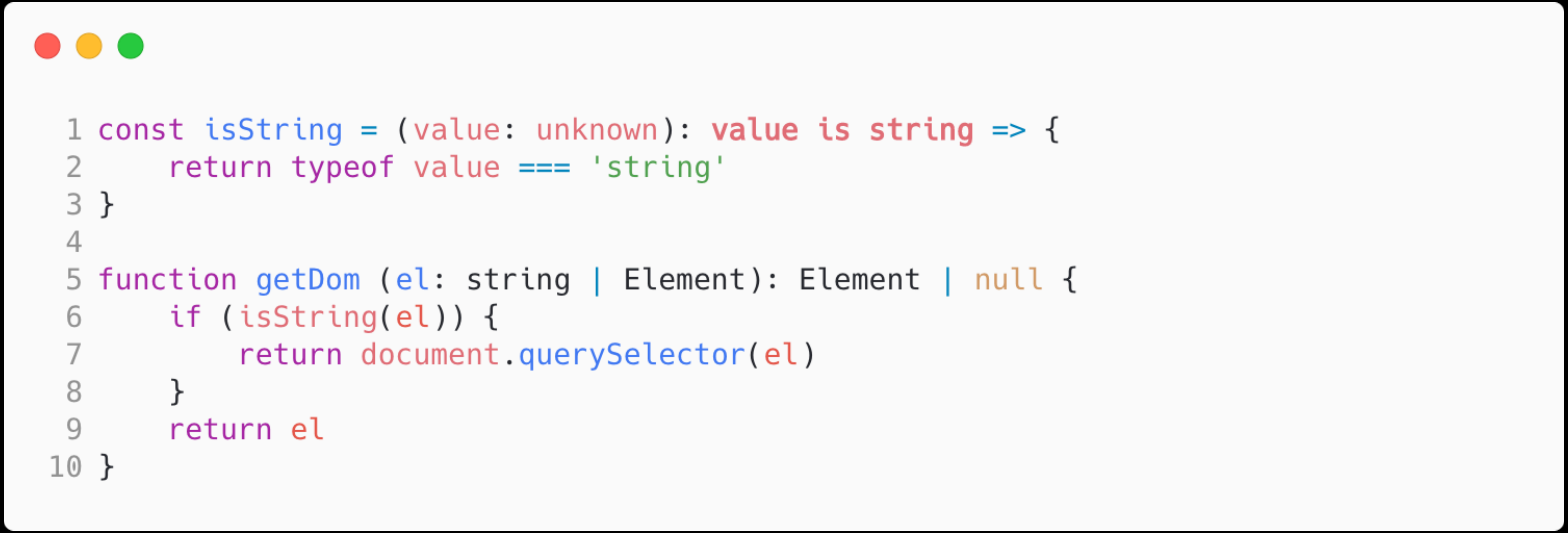
example 1 擅用类型推断



```
const isString = (value: unknown): boolean => {
    return typeof value === 'string'
}

function getDom (el: string | Element): Element | null {
    if (isString(el)) {
        return document.querySelector(el)
    }
    return el
}
```

如何自己书写泛型

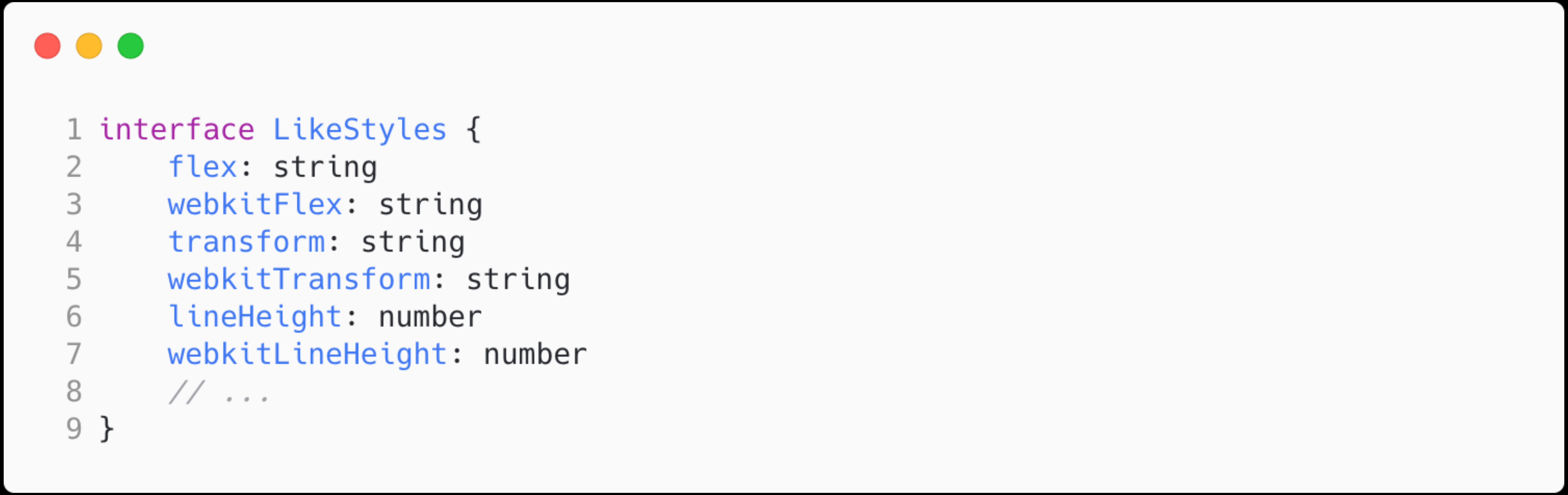
example 1 擅用类型推断



```
const isString = (value: unknown): value is string => {
    return typeof value === 'string'
}

function getDom (el: string | Element): Element | null {
    if (isString(el)) {
        return document.querySelector(el)
    }
    return el
}
```

如何自己书写泛型

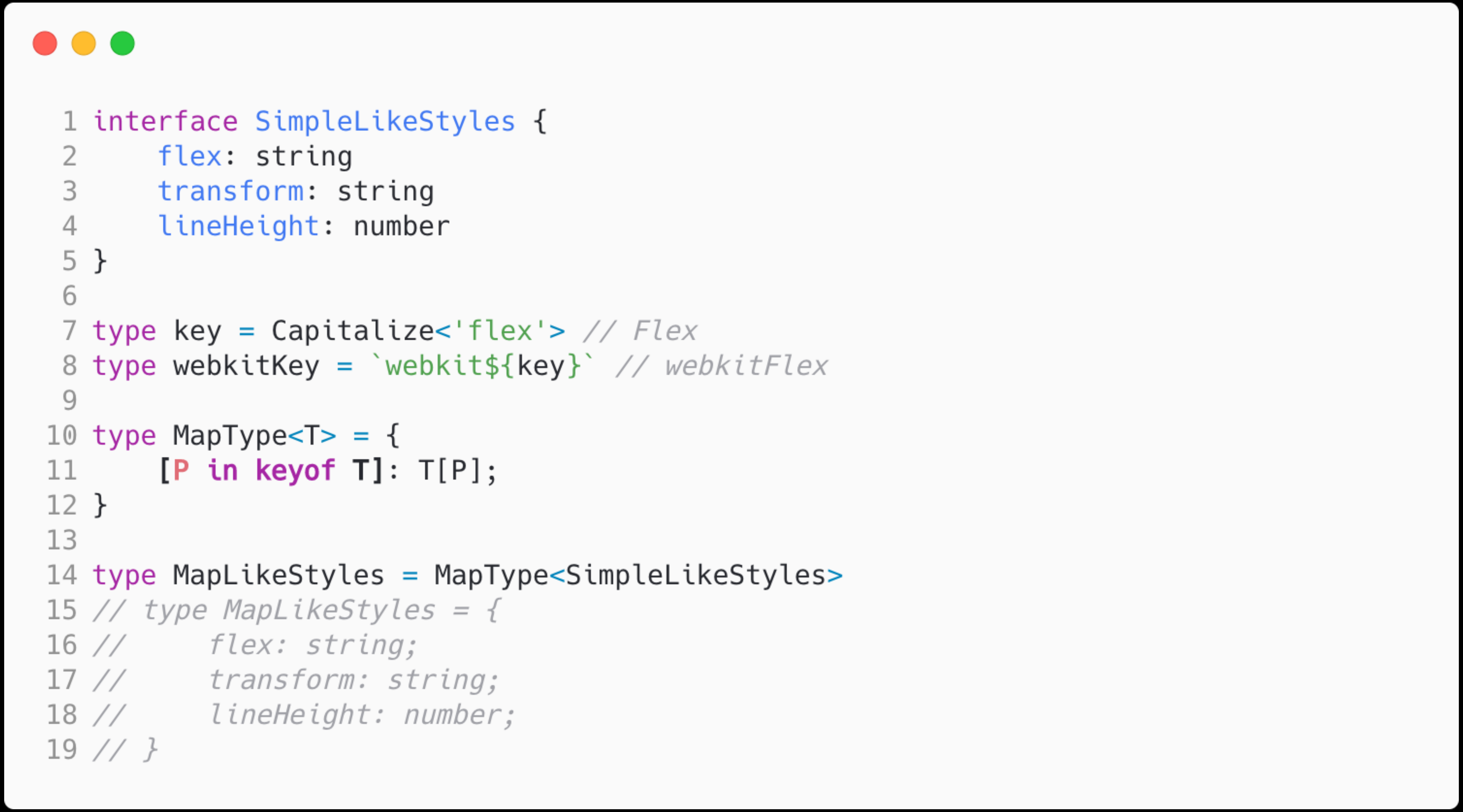
example 2 扩展新的泛型



```
interface LikeStyles {
    flex: string
    webkitFlex: string
    transform: string
    webkitTransform: string
    lineHeight: number
    webkitLineHeight: number
    // ...
}
```

如何自己书写泛型

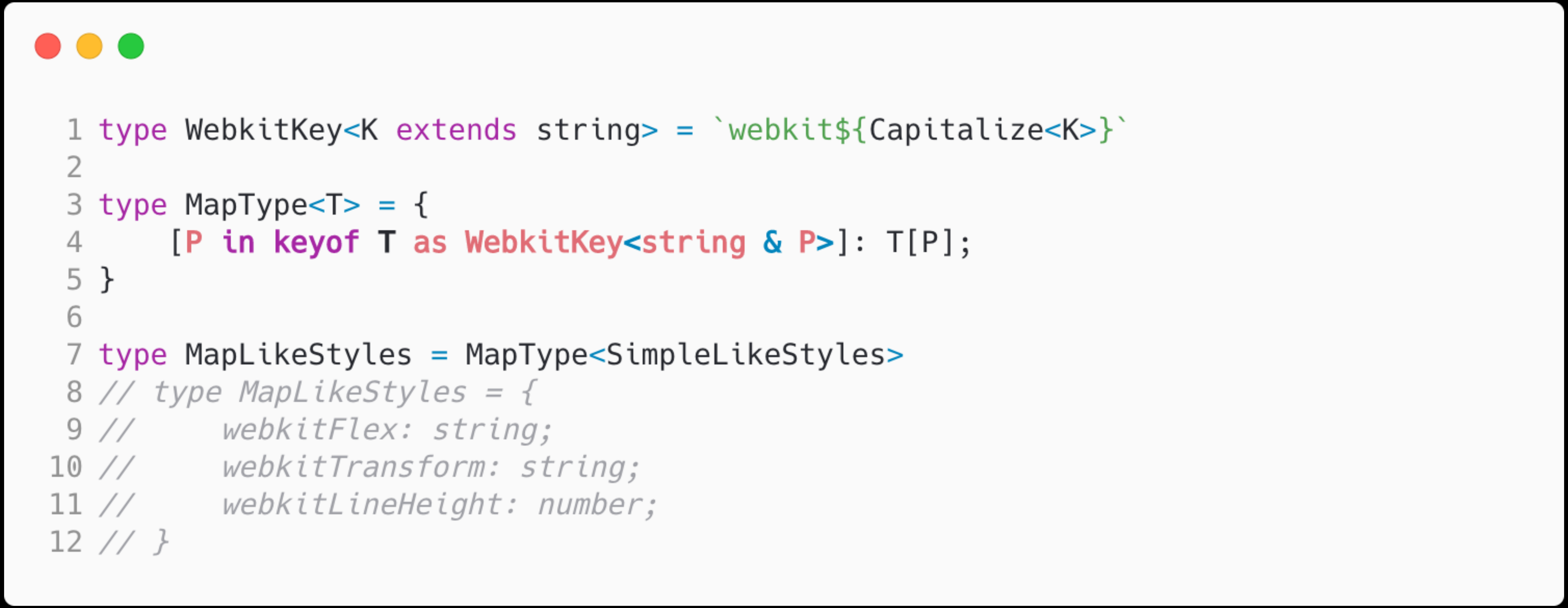
example 2 扩展新的泛型



```typescript
interface SimpleLikeStyles {
    flex: string
    transform: string
    lineHeight: number
}

type key = Capitalize<'flex'> // Flex
type webkitKey = `webkit${key}` // webkitFlex

type MapType<T> = {
    [P in keyof T]: T[P];
}

type MapLikeStyles = MapType<SimpleLikeStyles>
// type MapLikeStyles = {
//     flex: string;
//     transform: string;
//     lineHeight: number;
// }
```

如何自己书写泛型

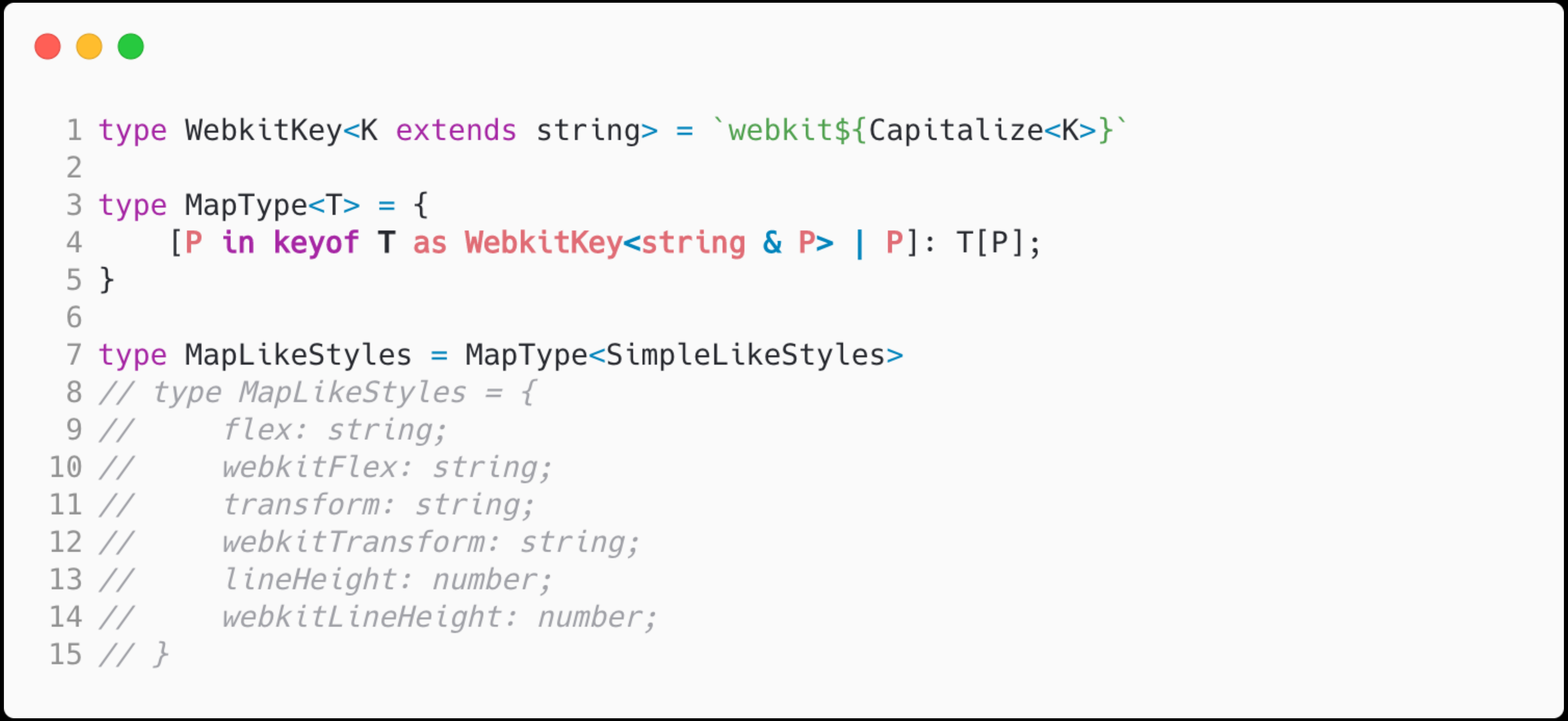
example 2 扩展新的泛型



```
type WebkitKey<K extends string> = `webkit${Capitalize<K>}`

type MapType<T> = {
    [P in keyof T as WebkitKey<string & P>]: T[P];
}

type MapLikeStyles = MapType<SimpleLikeStyles>
// type MapLikeStyles = {
//      webkitFlex: string;
//      webkitTransform: string;
//      webkitLineHeight: number;
// }
```

如何自己书写泛型

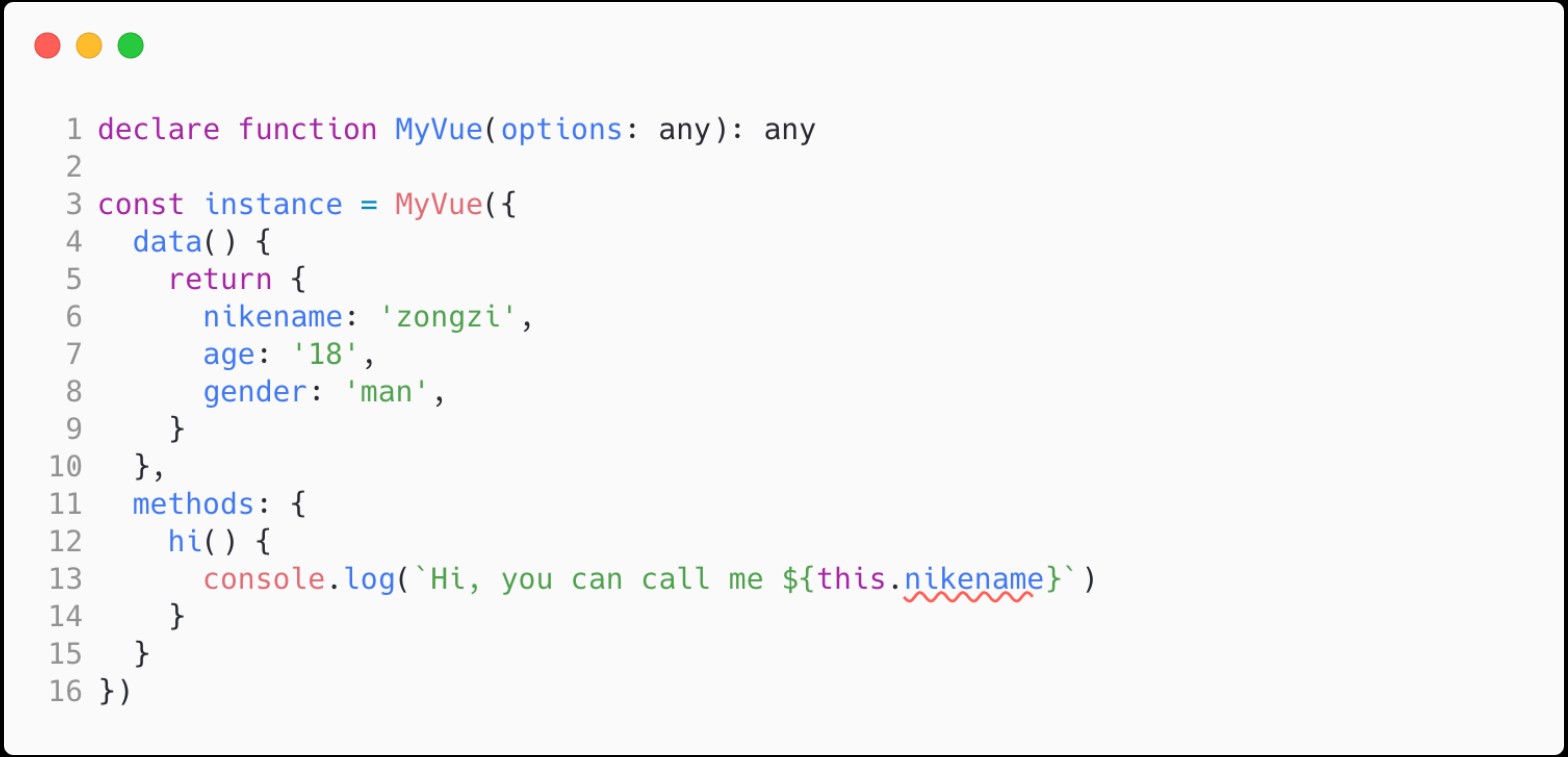
example 2 扩展新的泛型



```typescript
type WebkitKey<K extends string> = `webkit${Capitalize<K>}`

type MapType<T> = {
    [P in keyof T as WebkitKey<string & P> | P]: T[P];
}

type MapLikeStyles = MapType<SimpleLikeStyles>
// type MapLikeStyles = {
//       flex: string;
//       webkitFlex: string;
//       transform: string;
//       webkitTransform: string;
//       lineHeight: number;
//       webkitLineHeight: number;
// }
```

如何自己书写泛型

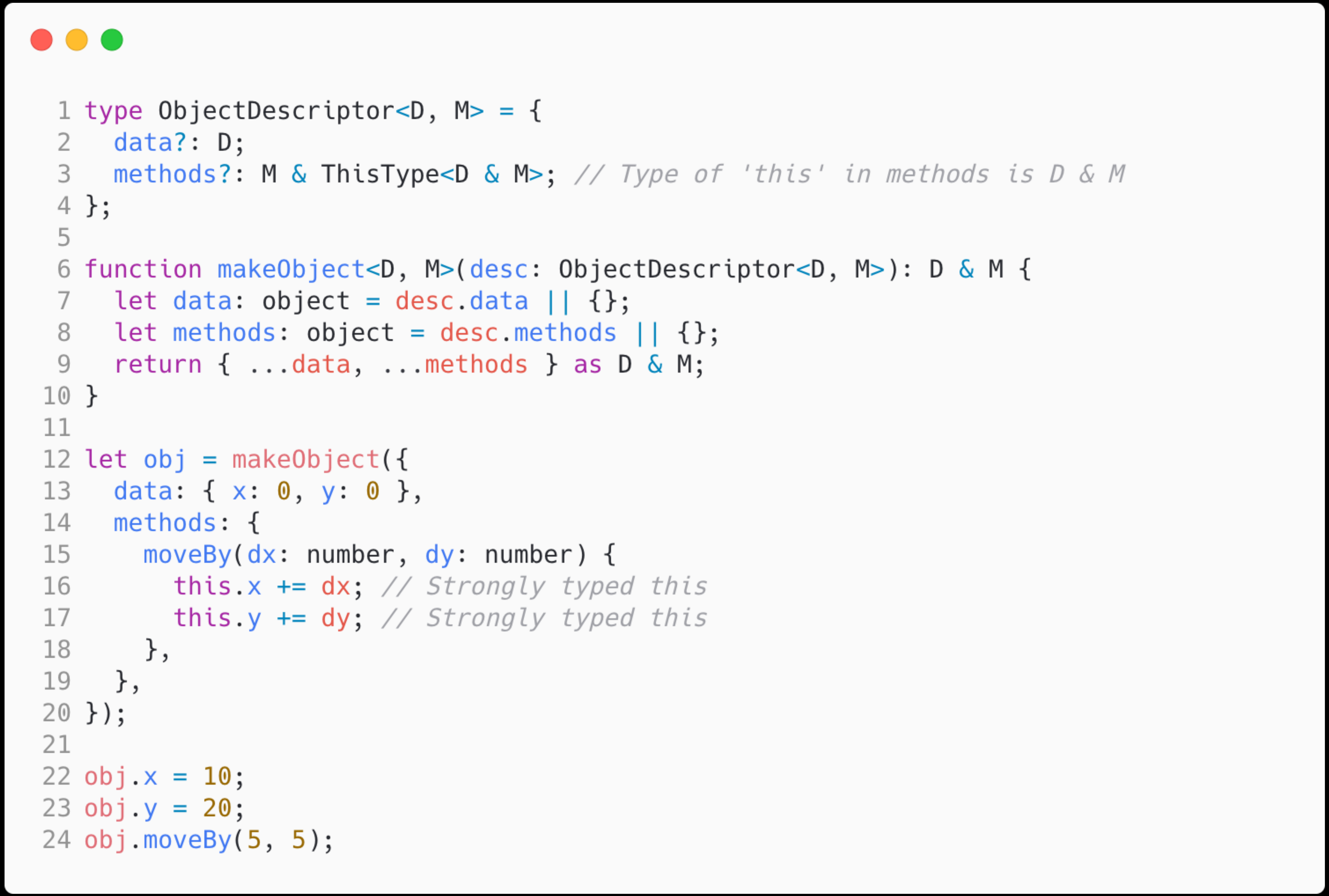
example 3 如何像 Vue 那样可以在 methods 中获得 data 的返回类型



```
declare function MyVue(options: any): any

const instance = MyVue({
  data() {
    return {
      nikename: 'zongzi',
      age: '18',
      gender: 'man',
    }
  },
  methods: {
    hi() {
      console.log(`Hi, you can call me ${this.nikename}`)
    }
  }
})
```

如何自己书写泛型

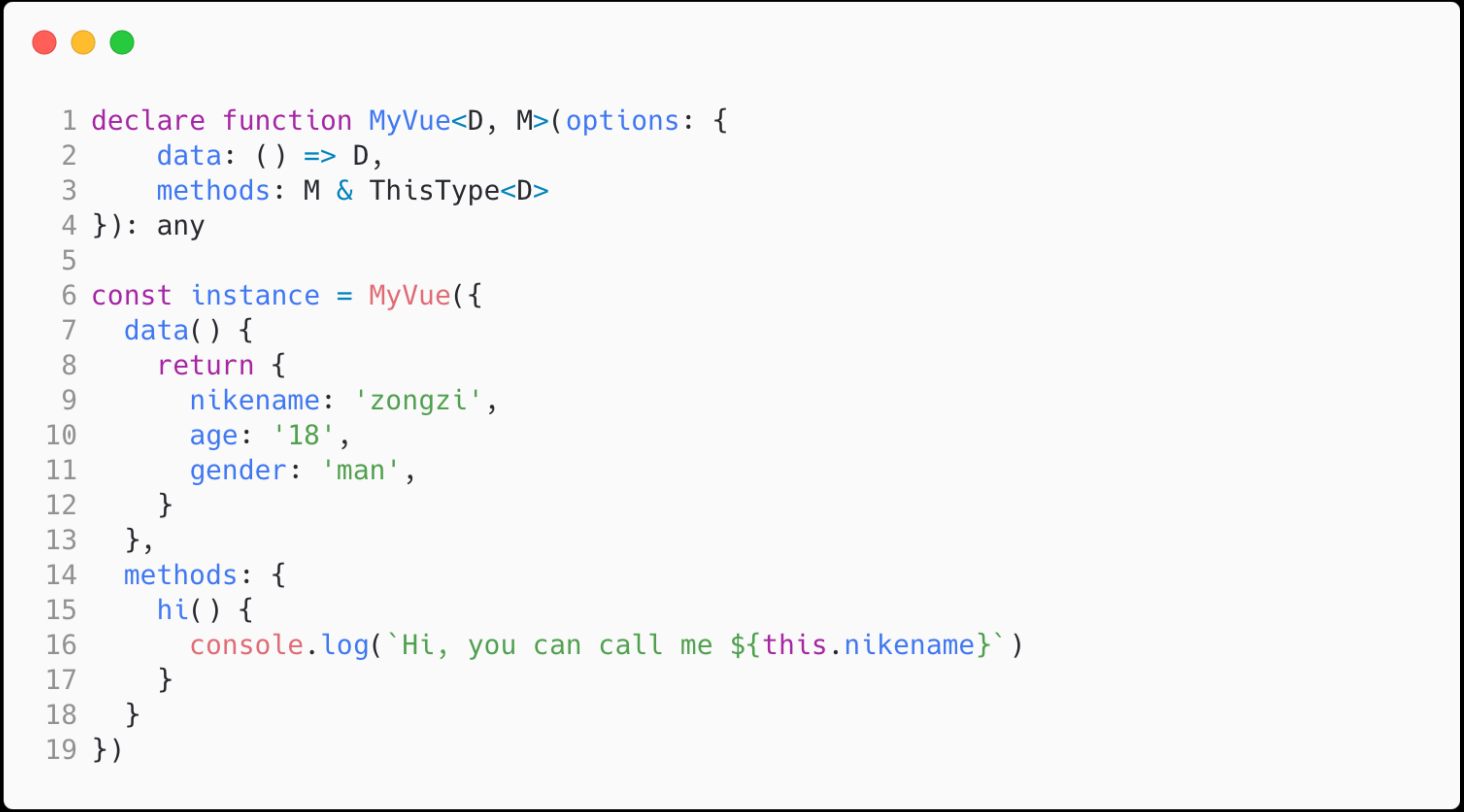
example 3 如何像 Vue 那样可以在 methods 中获得 data 的返回类型



```typescript
type ObjectDescriptor<D, M> = {
  data?: D;
  methods?: M & ThisType<D & M>; // Type of 'this' in methods is D & M
};

function makeObject<D, M>(desc: ObjectDescriptor<D, M>): D & M {
  let data: object = desc.data || {};
  let methods: object = desc.methods || {};
  return { ...data, ...methods } as D & M;
}

let obj = makeObject({
  data: { x: 0, y: 0 },
  methods: {
    moveBy(dx: number, dy: number) {
      this.x += dx; // Strongly typed this
      this.y += dy; // Strongly typed this
    },
  },
});

obj.x = 10;
obj.y = 20;
obj.moveBy(5, 5);
```

如何自己书写泛型

example 3 如何像 Vue 那样可以在 methods 中获得 data 的返回类型



```
declare function MyVue<D, M>(options: {
    data: () => D,
    methods: M & ThisType<D>
}): any

const instance = MyVue({
  data() {
    return {
      nikename: 'zongzi',
      age: '18',
      gender: 'man',
    }
  },
  methods: {
    hi() {
      console.log(`Hi, you can call me ${this.nikename}`)
    }
  }
})
```

提问环节

谢谢！