23 Jan 2023

TypeScript Basics

Aleksey Kozlenkov

Evolution Engineering | ACCELERATING EVOLUTION

What is TypeScript?

www.evolution.com

- TypeScript is a **typed superset** of JavaScript
- All valid JS code is a valid TS code (ommitting TS errors)
- TypeScript checks your code **statically** before it’s executed
- TypeScript code is transpiled to JavaScript code by TS compiler or Babel, all TS types are **erased before runtime**.
- TypeScript benefits:
    - code safety & stability;
    - easier refactoring;
    - features & comfort & saves time.

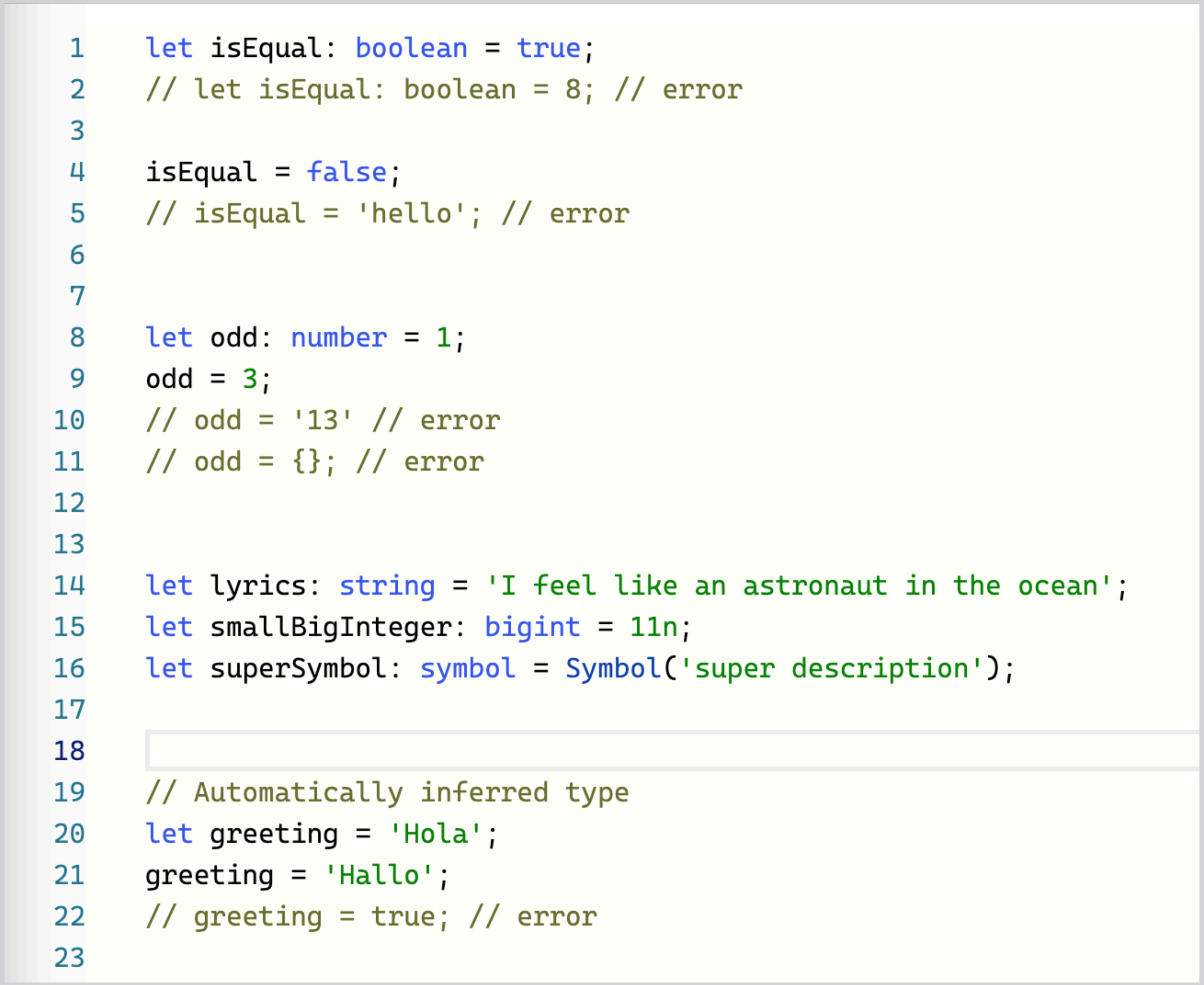
TS types vs JS types

JS	TS
null	null
undefined	undefined
boolean	boolean / Boolean
string	string / String
symbol	symbol / Symbol
number	number / Number
bigint	bigint
object	object / Object
function	function / Function
	any
	unknown
	Array / [] / Tuple
	void
	never
	enum

- Do not use Boolean/String/Number/Object/Symbol/Function in TS as types.
- Object vs object
- Avoid **any** unless it is strictly necessary

Evolution Engineering | ACCELERATING EVOLUTION

Types: primitives

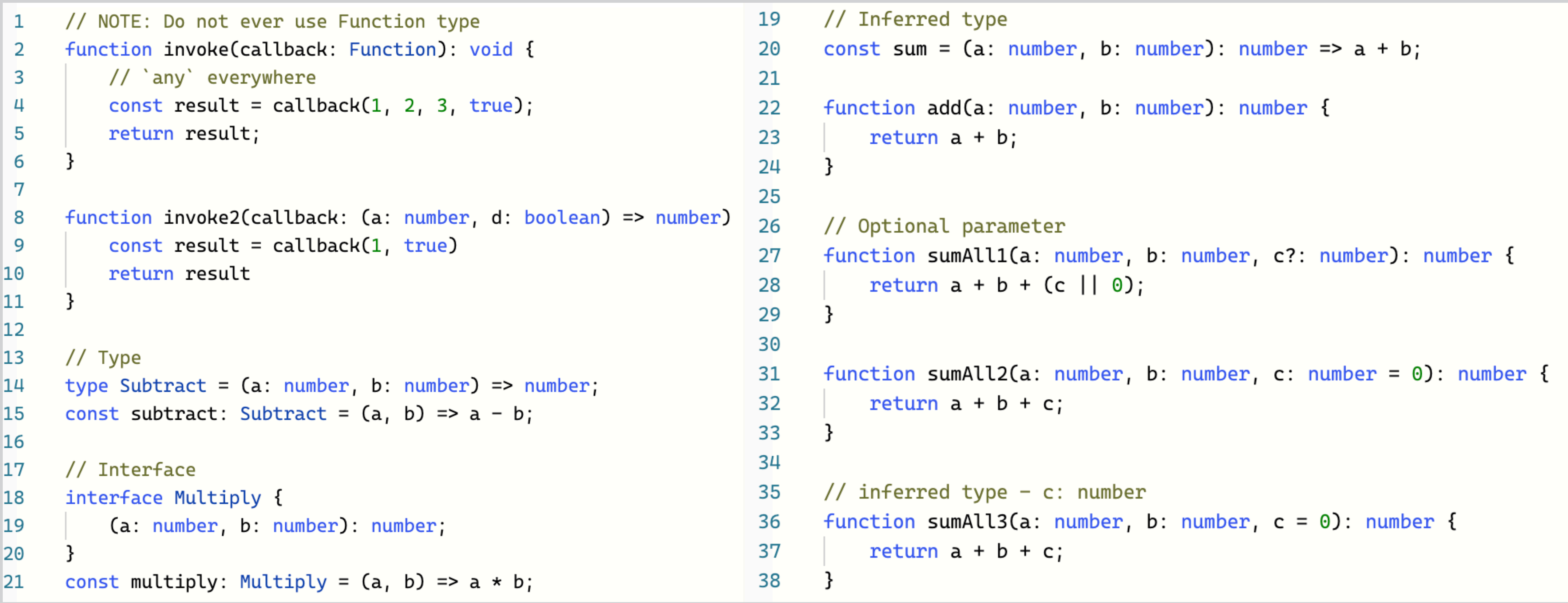


```
let isEqual: boolean = true;
// let isEqual: boolean = 8; // error

isEqual = false;
// isEqual = 'hello'; // error


let odd: number = 1;
odd = 3;
// odd = '13' // error
// odd = {}; // error


let lyrics: string = 'I feel like an astronaut in the ocean';
let smallBigInteger: bigint = 11n;
let superSymbol: symbol = Symbol('super description');


// Automatically inferred type
let greeting = 'Hola';
greeting = 'Hallo';
// greeting = true; // error

```

- Playground link

Types: function

- Playground link



```typescript
// NOTE: Do not ever use Function type
function invoke(callback: Function): void {
    // `any` everywhere
    const result = callback(1, 2, 3, true);
    return result;
}

function invoke2(callback: (a: number, d: boolean) => number)
    const result = callback(1, true)
    return result
}

// Type
type Subtract = (a: number, b: number) => number;
const subtract: Subtract = (a, b) => a - b;

// Interface
interface Multiply {
    (a: number, b: number): number;
}
const multiply: Multiply = (a, b) => a * b;
```

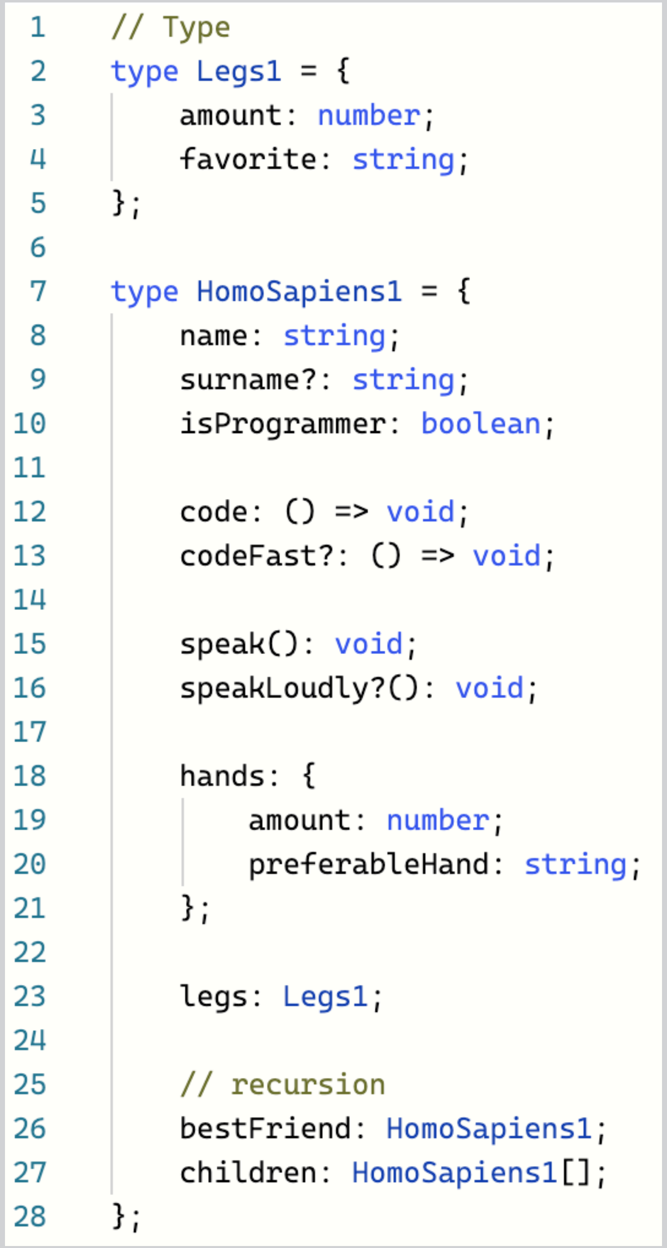
```typescript
// Inferred type
const sum = (a: number, b: number): number => a + b;

function add(a: number, b: number): number {
    return a + b;
}

// Optional parameter
function sumAll1(a: number, b: number, c?: number): number {
    return a + b + (c || 0);
}

function sumAll2(a: number, b: number, c: number = 0): number {
    return a + b + c;
}

// inferred type - c: number
function sumAll3(a: number, b: number, c = 0): number {
    return a + b + c;
}
```

Evolution Engineering | ACCELERATING EVOLUTION

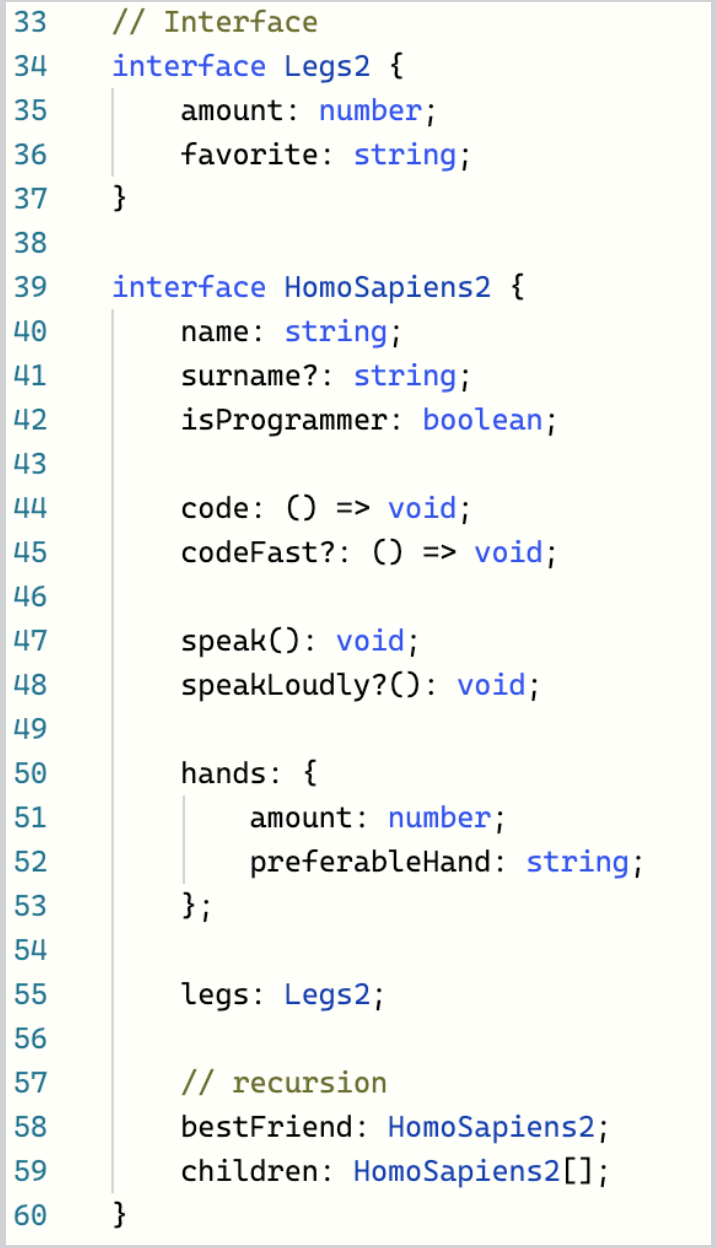
Types: function

- Function excercise

Types: object

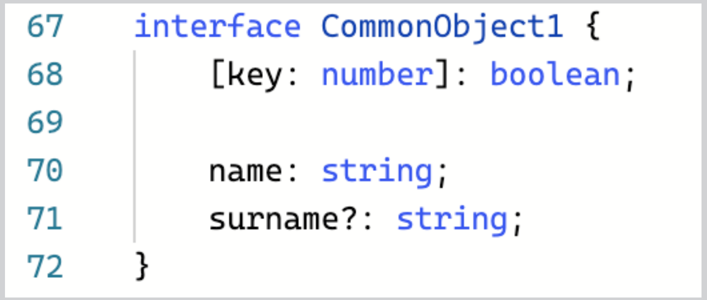


```typescript
// Type
type Legs1 = {
    amount: number;
    favorite: string;
};

type HomoSapiens1 = {
    name: string;
    surname?: string;
    isProgrammer: boolean;

    code: () => void;
    codeFast?: () => void;

    speak(): void;
    speakLoudly?(): void;

    hands: {
        amount: number;
        preferableHand: string;
    };

    legs: Legs1;

    // recursion
    bestFriend: HomoSapiens1;
    children: HomoSapiens1[];
};
```

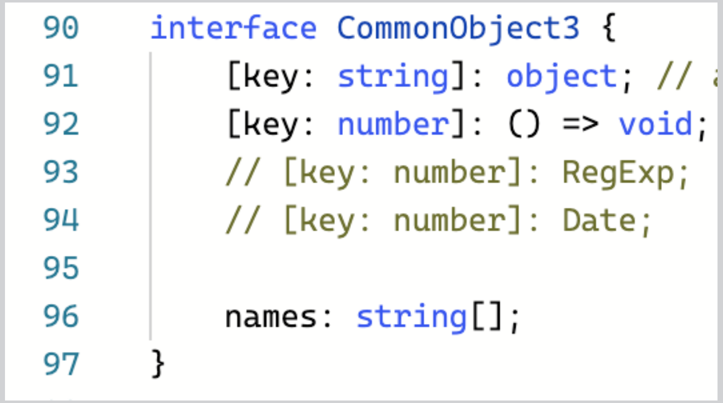


```typescript
// Interface
interface Legs2 {
    amount: number;
    favorite: string;
}

interface HomoSapiens2 {
    name: string;
    surname?: string;
    isProgrammer: boolean;

    code: () => void;
    codeFast?: () => void;

    speak(): void;
    speakLoudly?(): void;

    hands: {
        amount: number;
        preferableHand: string;
    };

    legs: Legs2;

    // recursion
    bestFriend: HomoSapiens2;
    children: HomoSapiens2[];
}
```

- Playground
- Types vs interface



```typescript
interface CommonObject1 {
    [key: number]: boolean;

    name: string;
    surname?: string;
}
```



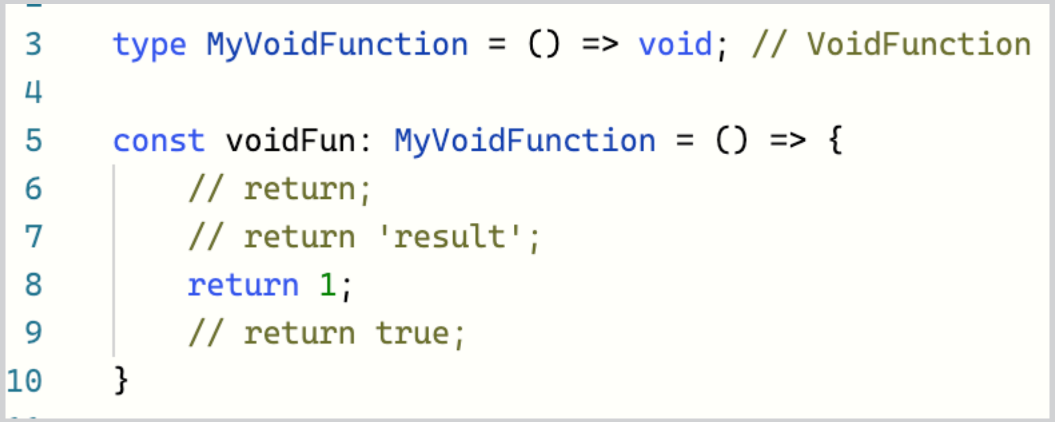
```typescript
interface CommonObject3 {
    [key: string]: object; //
    [key: number]: () => void;
    // [key: number]: RegExp;
    // [key: number]: Date;

    names: string[];
}
```

Evolution Engineering | ACCELERATING EVOLUTION

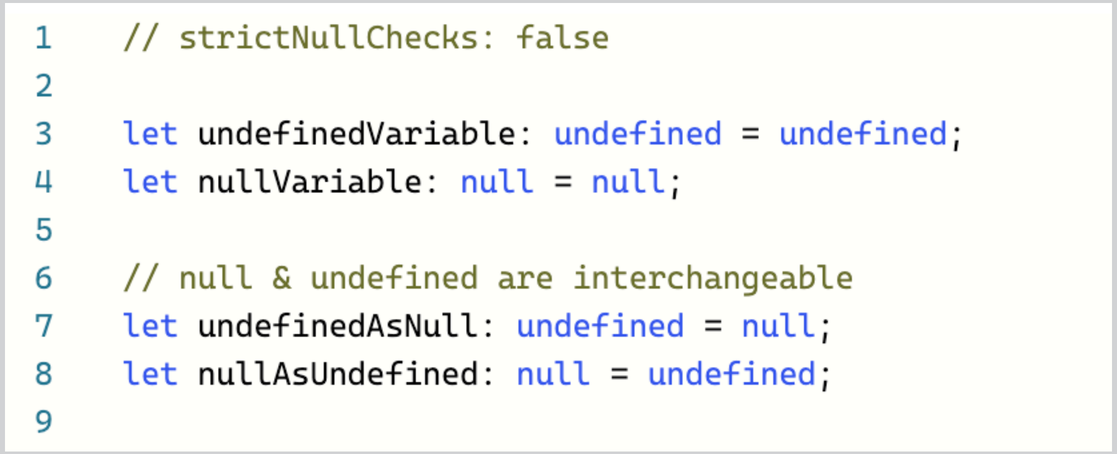
Types: undefined, void, null

www.evolution.com

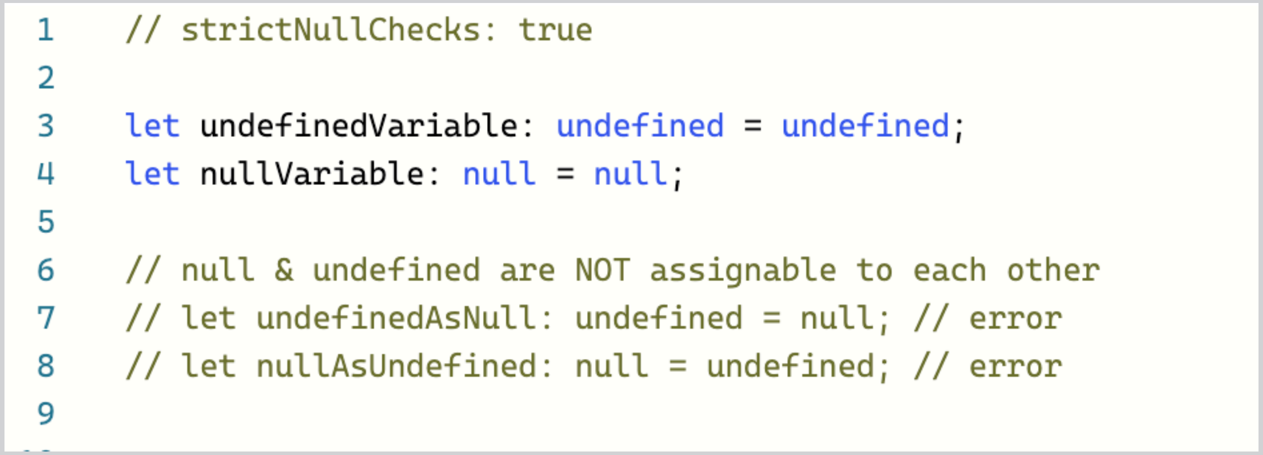
- [Playground: strictNullChecks: false]()
- [Playground: strictNullChecks: true]()
- [Playground: void]()



```typescript
type MyVoidFunction = () => void; // VoidFunction

const voidFun: MyVoidFunction = () => {
    // return;
    // return 'result';
    return 1;
    // return true;
}
```



```typescript
// strictNullChecks: false

let undefinedVariable: undefined = undefined;
let nullVariable: null = null;

// null & undefined are interchangeable
let undefinedAsNull: undefined = null;
let nullAsUndefined: null = undefined;

```

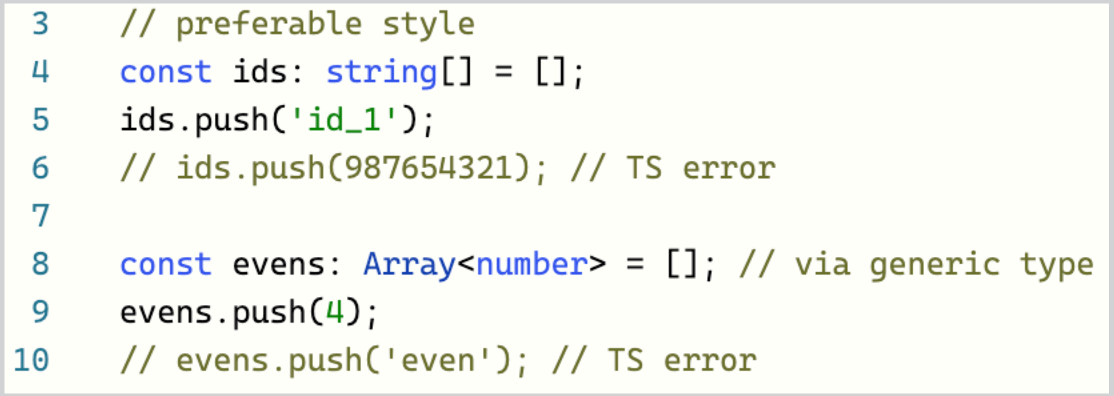


```typescript
// strictNullChecks: true

let undefinedVariable: undefined = undefined;
let nullVariable: null = null;

// null & undefined are NOT assignable to each other
// let undefinedAsNull: undefined = null; // error
// let nullAsUndefined: null = undefined; // error

```

Evolution Engineering | ACCELERATING EVOLUTION

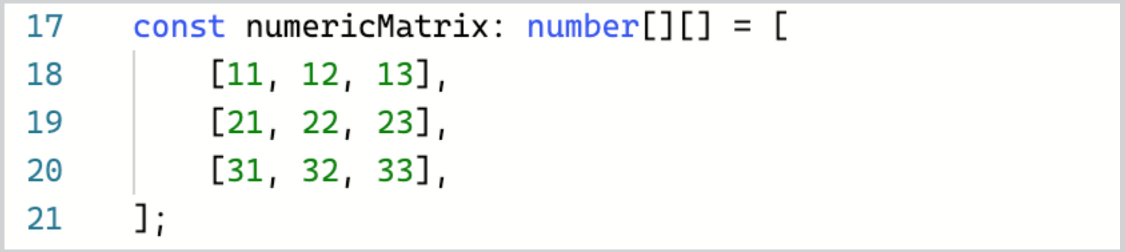
Types: Array / []

www.evolution.com

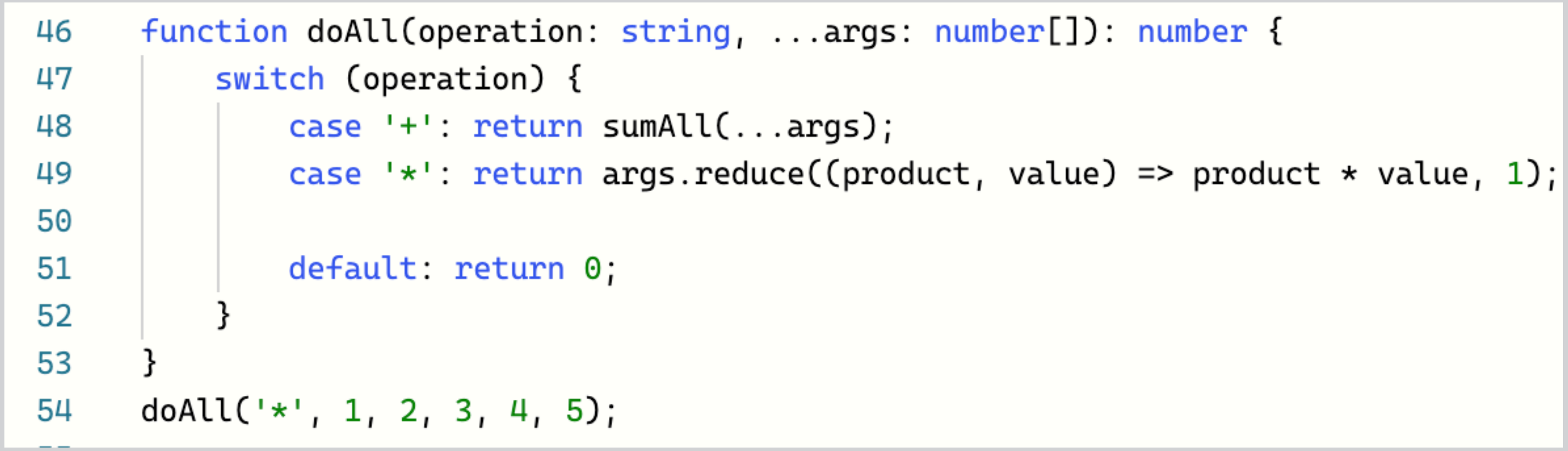


```typescript
// preferable style
const ids: string[] = [];
ids.push('id_1');
// ids.push(987654321); // TS error

const evens: Array<number> = []; // via generic type
evens.push(4);
// evens.push('even'); // TS error
```

- Plaground link



```typescript
const numericMatrix: number[][] = [
    [11, 12, 13],
    [21, 22, 23],
    [31, 32, 33],
];
```

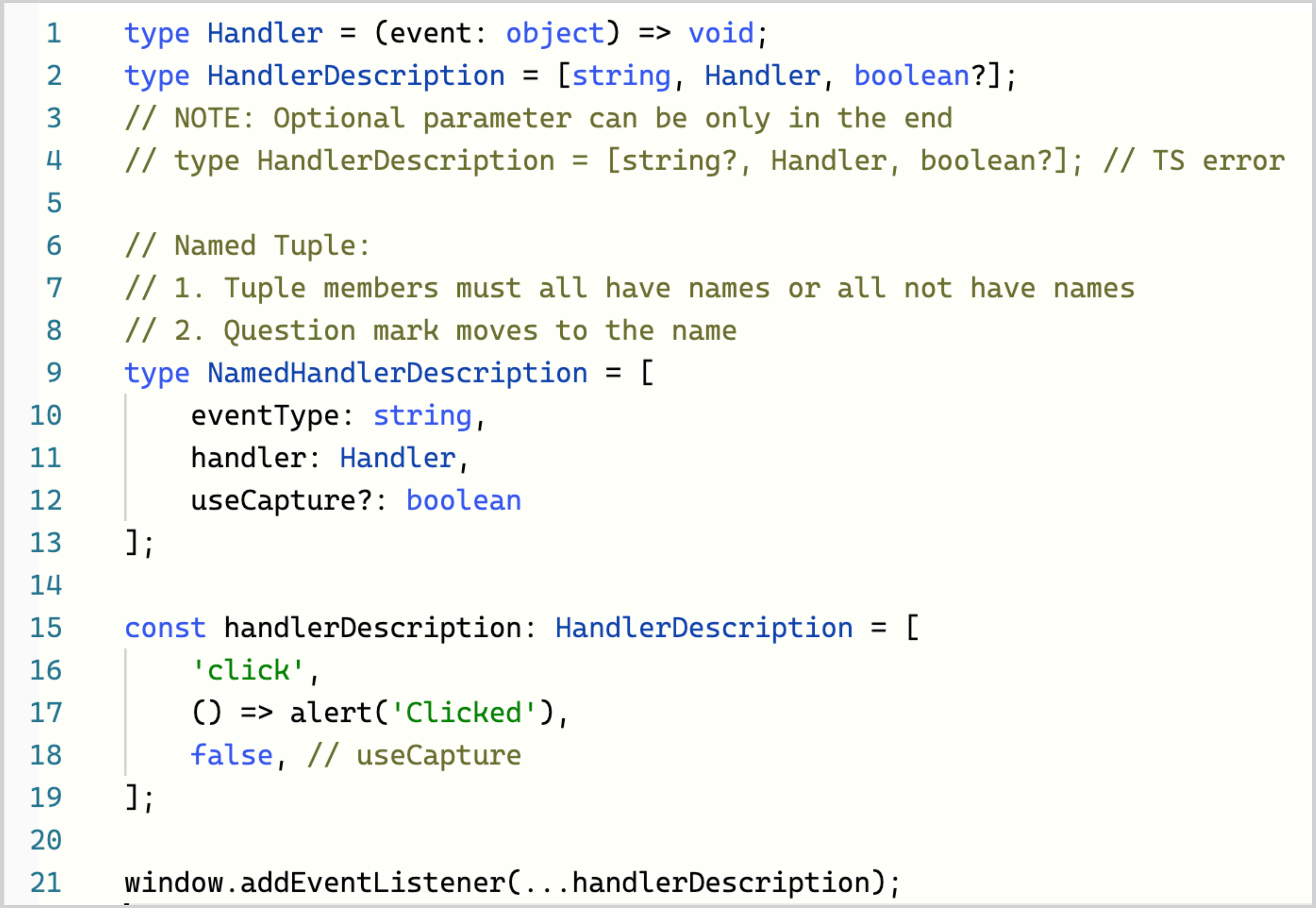


```typescript
function doAll(operation: string, ...args: number[]): number {
    switch (operation) {
        case '+': return sumAll(...args);
        case '*': return args.reduce((product, value) => product * value, 1);

        default: return 0;
    }
}
doAll('*', 1, 2, 3, 4, 5);
```

Evolution Engineering | ACCELERATING EVOLUTION

Types: Tuple

www.evolution.com



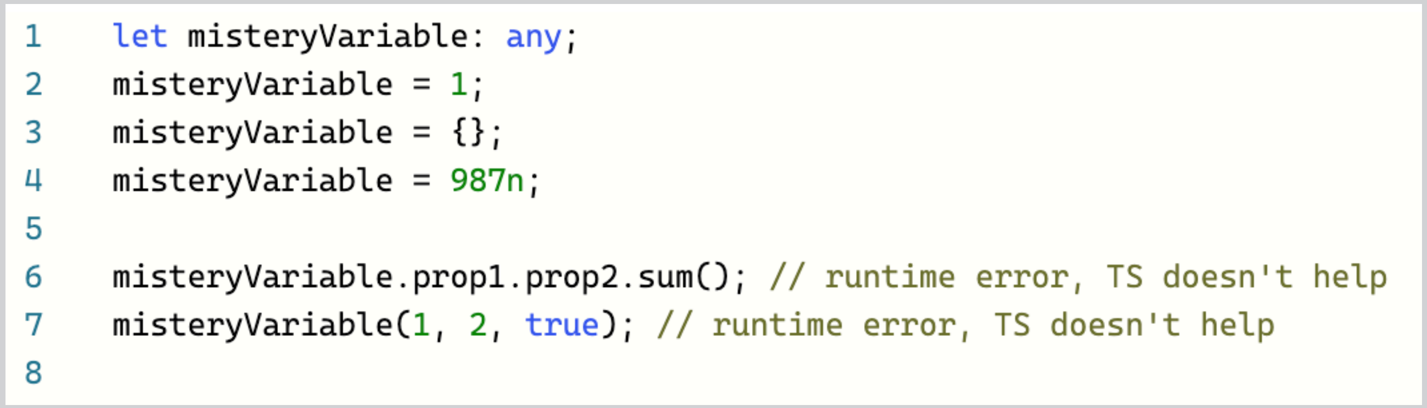
```
type Handler = (event: object) => void;
type HandlerDescription = [string, Handler, boolean?];
// NOTE: Optional parameter can be only in the end
// type HandlerDescription = [string?, Handler, boolean?]; // TS error

// Named Tuple:
// 1. Tuple members must all have names or all not have names
// 2. Question mark moves to the name
type NamedHandlerDescription = [
    eventType: string,
    handler: Handler,
    useCapture?: boolean
];

const handlerDescription: HandlerDescription = [
    'click',
    () => alert('Clicked'),
    false, // useCapture
];

window.addEventListener(...handlerDescription);
```

- [Playground link]

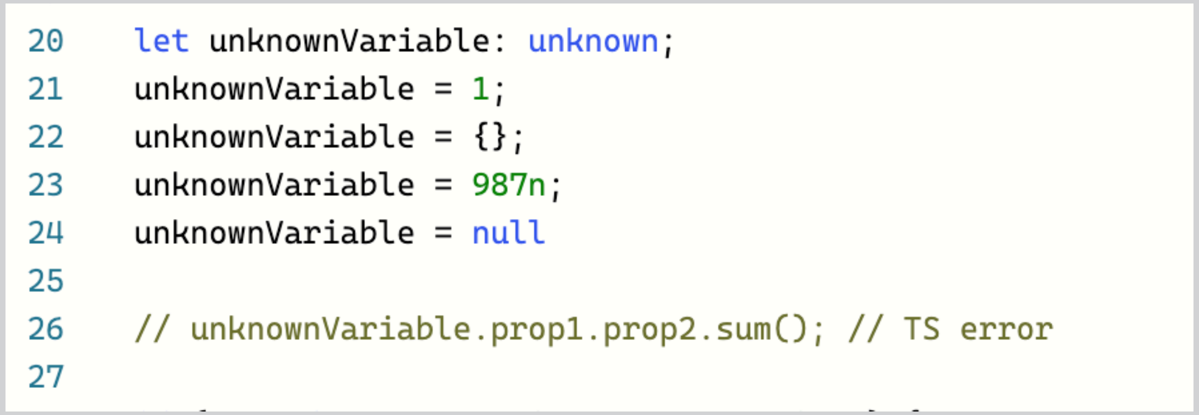
Types: object/arrays

- Object/arrays excersize

Types: any, unknown



```typescript
let misteryVariable: any;
misteryVariable = 1;
misteryVariable = {};
misteryVariable = 987n;

misteryVariable.prop1.prop2.sum(); // runtime error, TS doesn't help
misteryVariable(1, 2, true); // runtime error, TS doesn't help

```



```typescript
let unknownVariable: unknown;
unknownVariable = 1;
unknownVariable = {};
unknownVariable = 987n;
unknownVariable = null

// unknownVariable.prop1.prop2.sum(); // TS error

```

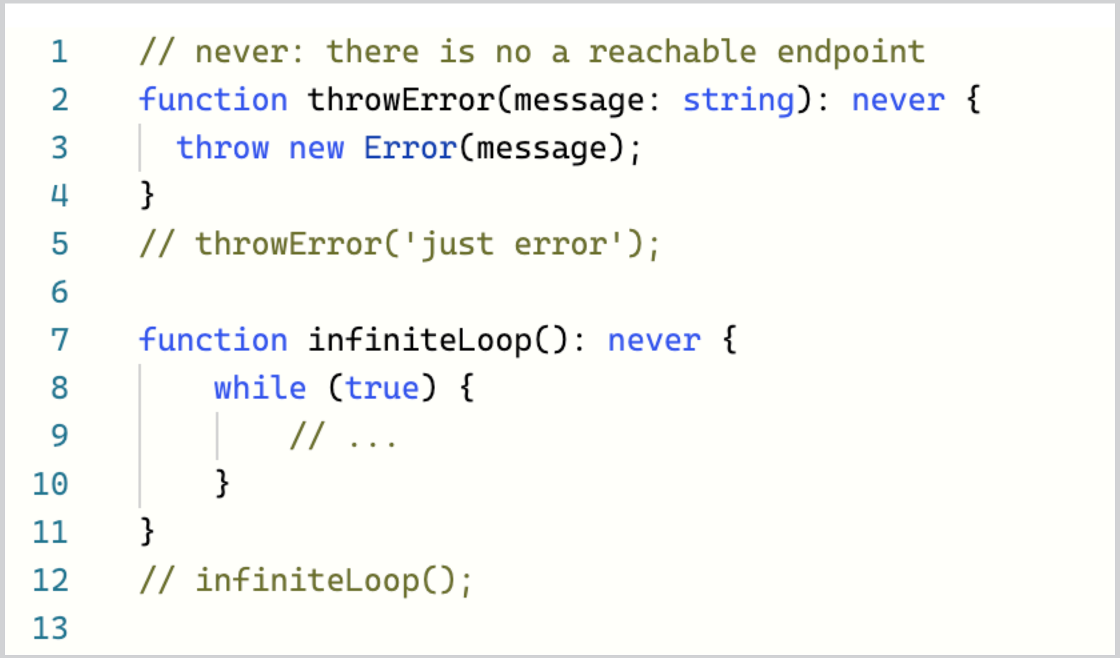
- Playground link

Evolution Engineering | ACCELERATING EVOLUTION

Types: never

www.evolution.com

- Playground link

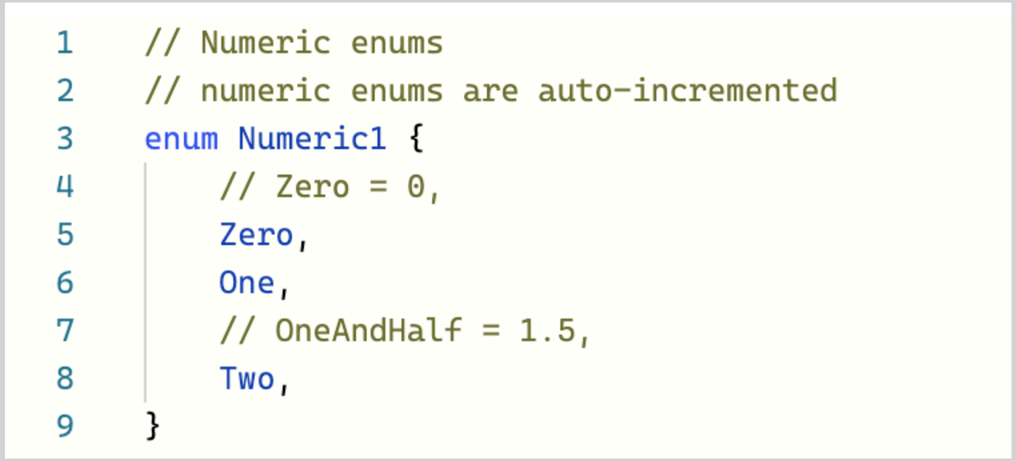


```
// never: there is no a reachable endpoint
function throwError(message: string): never {
  throw new Error(message);
}
// throwError('just error');

function infiniteLoop(): never {
    while (true) {
        // ...
    }
}
// infiniteLoop();

```

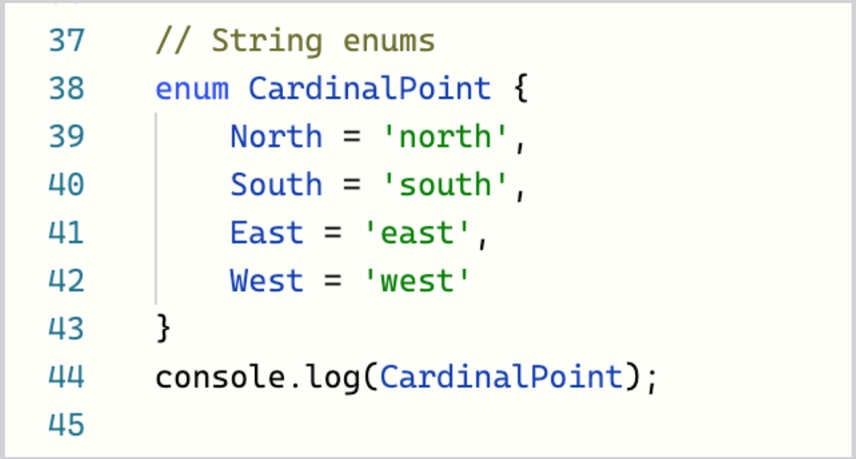
Evolution Engineering | ACCELERATING EVOLUTION

Enum

www.evolution.com



```
// Numeric enums
// numeric enums are auto-incremented
enum Numeric1 {
    // Zero = 0,
    Zero,
    One,
    // OneAndHalf = 1.5,
    Two,
}
```



```
// String enums
enum CardinalPoint {
    North = 'north',
    South = 'south',
    East = 'east',
    West = 'west'
}
console.log(CardinalPoint);
```

- Playground link
- More on enum vs object [1] [2]

Evolution Engineering | ACCELERATING EVOLUTION

www.evolution.com