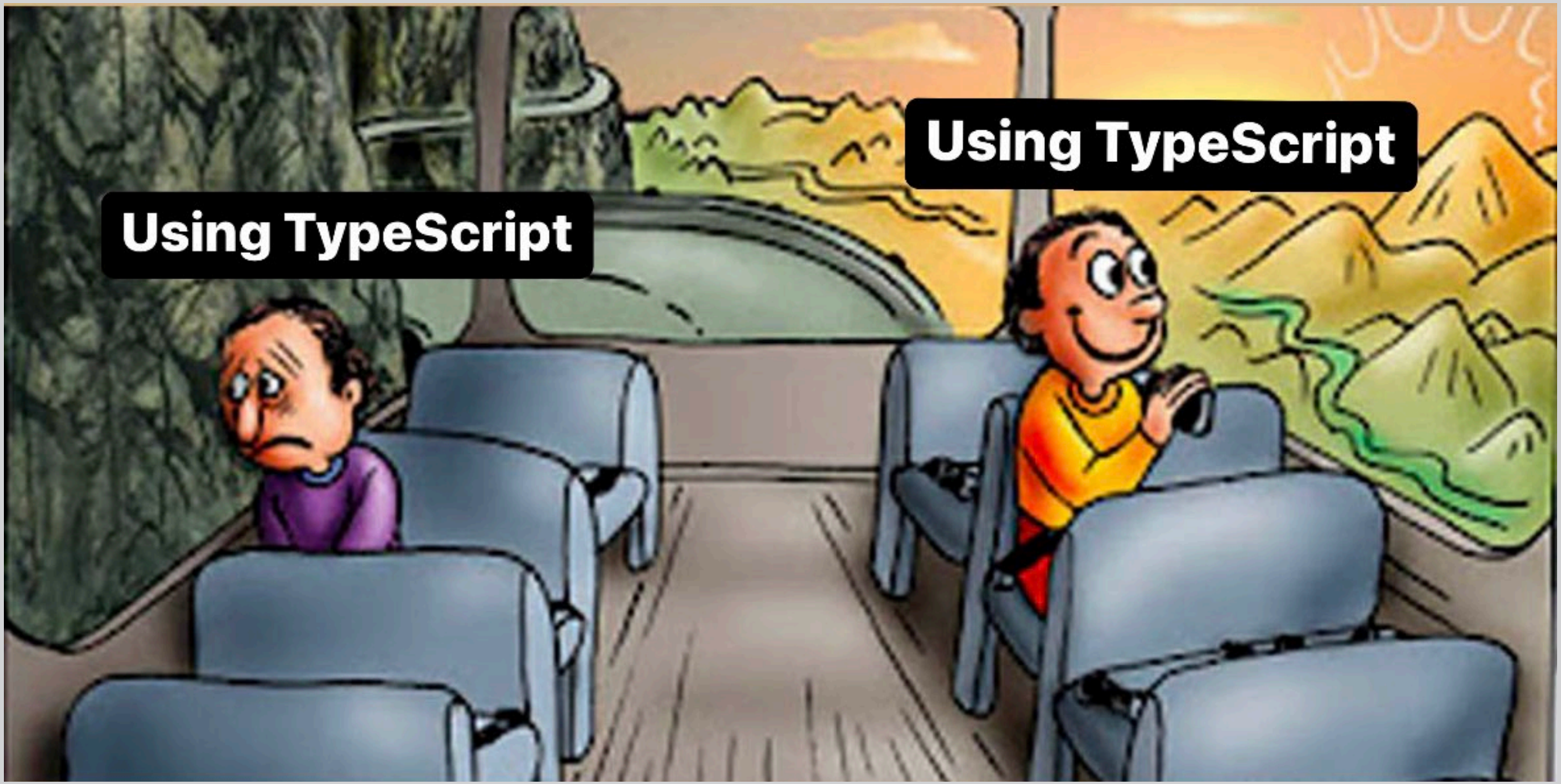
30 Jan 2023

TypeScript 2

Aleksey Kozlenkov


Using TypeScript
Using TypeScript

Union & Intersection


```ts
// Union Types
type Id = number | string;

function testId(id: Id) {
    id.toLocaleString()
    // id.charAt(0) // ts error
    // id.toFixed() // ts error
}

function convert(id: Id): boolean | symbol {
    if (typeof id === 'string') {
        return Symbol(id);
    }

    return !!id;
}
```

- Playground link

```ts
// Intersection Types
type Superman = Human & Animal;
const superman: Superman = {
    work() { },
    relax() { },
};
superman.relax();
superman.work();

```

Literal types & *as const*

- Playground link

```ts
let anyStringValue = 'Hey Bob' // string
const TS_URL = 'https://www.typescriptlang.org/';
const PI = 3.14;
```

```ts
interface ImmutableObject {
    readonly do: () => void;
    readonly id: number;
    readonly obj: {
        a: string;
        b: string;
    };
}
```

```ts
const AGE_BY_NAME = {
    Name1: 11,
    Name2: 32,
    Groznyi: {
        title: 'King',
        age: Infinity,
    },
} as const;
```

Inferred array type


```
3   const STRINGS = ['west', 'east']; // string[]
4   const STRINGS_NUMBERS = ['west', 'east', 450]; // (string | number)[]
5
6   function exists(
7       array: number[][],
8       value: number,
9   // ) {
10  ): [
11      repeatCount: number,
12      found: boolean,
13      coordinates: [column: number, row: number][],
14  ] {
15      let repeatCount = 0;
16      const coordinates: [column: number, row: number][] = [];
17
18      array.forEach((row, rowIndex) => {
19          row.forEach((currentValue, columnIndex) => {
20              if (currentValue === value) {
21                  repeatCount++;
22                  coordinates.push([rowIndex, columnIndex]);
23              }
24          });
25      });
26
27      return [repeatCount, !!repeatCount, coordinates];
```

- Playground link

Exhaustive check


```ts
enum CardinalPoint {
  North = 'north',
  South = 'south',
  East = 'east',
  West = 'west'
}

function assertNever(arg: never): never {
  throw new Error(`Unexpected argument: ${arg}`);
}

function getCoordinate(cardinalPoint: CardinalPoint): [number, number] {
  switch (cardinalPoint) {
    case CardinalPoint.North:
      return [0, 0];
    case CardinalPoint.South:
      return [1, 1];
    case CardinalPoint.East:
      return [0, 1];
    case CardinalPoint.West:
      return [1, 0];

    default:
      return assertNever(cardinalPoint);
  }
}
```

- Playground link

Clicky-clicky time

- Types excersise

Type indexed access


```typescript
interface Legs {
    amount: number;
    favorite: string;
}

interface HomoSapiens {
    name: string;
    surname?: string;
    isProgrammer: boolean;
    legs: Legs;

    move(x: string, y: string): string;
    speak(): void;

    bestFriend: HomoSapiens;
    children: HomoSapiens[];
}

type HomoSapiensName = HomoSapiens["name"];
type HomoSapiensSurname = HomoSapiens["surname"];
type HomoSapiensBestFriend = HomoSapiens["bestFriend"];

```

- Playground link

Type vs interface


```ts
// Type Alias
// A type alias is exactly that - a name for any type
// (objects, functions, primitives, unions, intersections, etc.)
type Value1 = string | number | boolean;
type Value2 = 'str' | 200 | true;
type Value3 = string;

type SomeFunction1 = {
    (index: number, code: string): string;
};
type SomeFunction2 = (index: number, code: string) => string;

```

```ts
// Interface
// An interface declaration is another way to name an object type
// ONLY OBJECTS
interface SomeFunctionInterface1 {
    (index: number, code: string): string;
}

interface HomoSapiensInterface {
    name: string;
}

interface SuperHomoSapiensInterface extends HomoSapiensInterface {
    isOk: boolean;
}

```

- Playground link

Evolution Engineering | ACCELERATING EVOLUTION

Thank you!

www.evolution.com