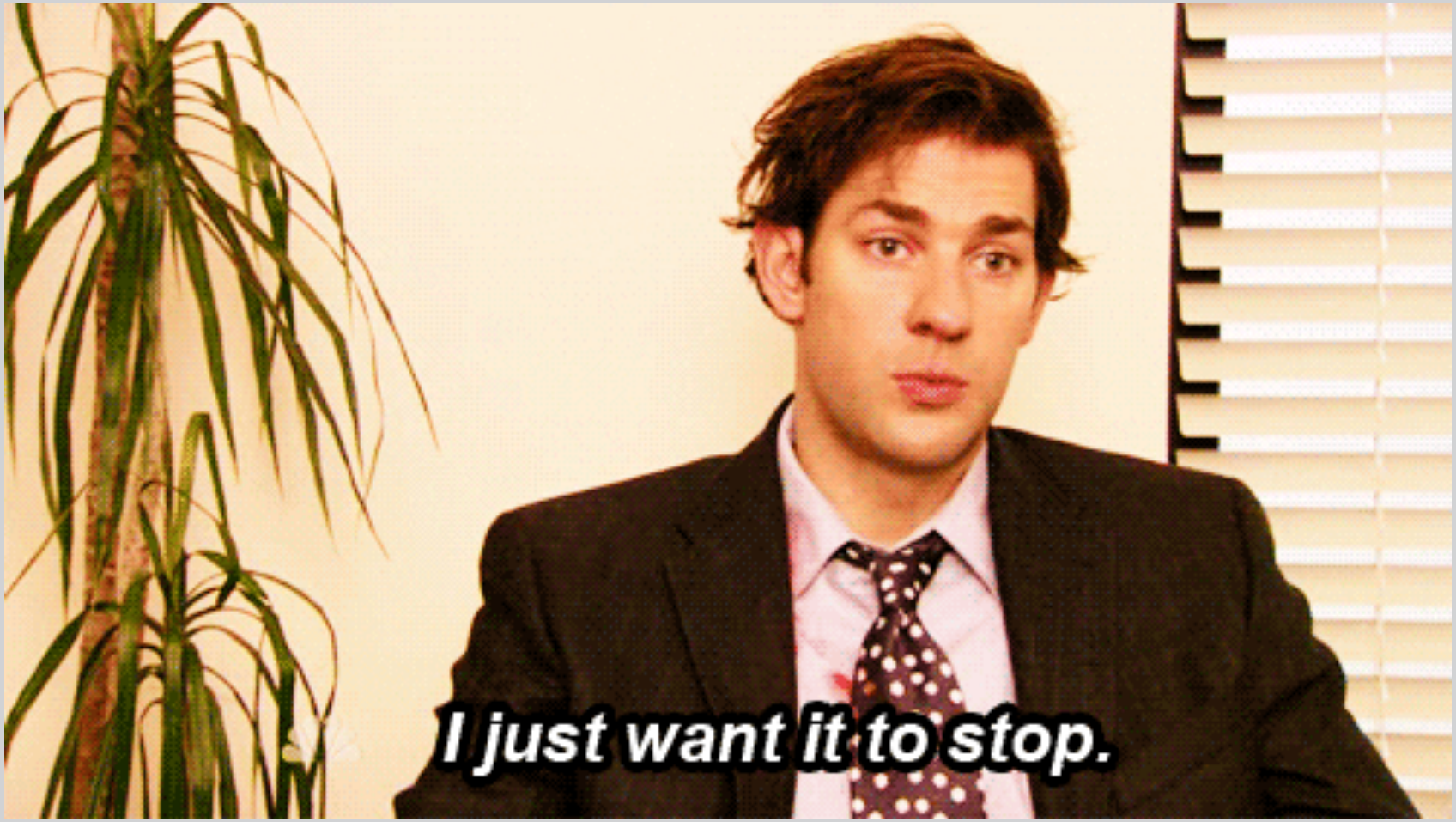
08 Feb 2023

TypeScript 3

Aleksey Kozlenkov


I just want it to stop.

Classes


```
/**
 *
 *  Simple Class
 *
 */
class Office {
    branch: string;
    employees: string[];

    constructor(branch: string) {
        this.branch = branch;
        this.employees = []
    }

    greet(): string {
        return `Hi, this is ${this.branch}`
    }
}

const paperCompany = new Office('White Paper')
console.log(paperCompany.greet())
```

- Playground link 1
- Playground link 2

Generics


```
7   function createArray<T>(value: T, length: number): T[] {
8       return new Array(length).fill(value);
9   }
10  const arr1 = createArray(123, 10);          // numbers
11  const arr2 = createArray('str', 10);        // strings
12  const arr3 = createArray(true, 10);         // booleans
13
```

```
1   class IDKeeper<Type> {
2     constructor(
3       public id: Type,
4     ) {}
5
6     public setNewId(id: Type): void {
7       this.id = id;
8     }
9   }
10
```

- Generics playground
- Generic classes playground

Utility types

- Playground link

```ts
// Awaited<Promise<Type>>
type PromiseResult1 = Awaited<Promise<string>>;
type PromiseResult2 = Awaited<boolean | Promise<string>>;
type PromiseResult3 = Awaited<Promise<Promise<Promise<string>>>>;


// Readonly<Type> / ReadonlyArray<Type>
type MyArray = number[];
type MyReadonlyObj = Readonly<{ prop: string; }>;
type MyReadonlyArray1 = Readonly<number[]>;


// NonNullable<Type>
type NonNullable1 = NonNullable<string | number | undefined>;
type NonNullable2 = NonNullable<symbol | null | [undefined]>;
type NonNullable3 = NonNullable<{ a(): void; } | false>;

```

Type assertion


```
7   const data = JSON.parse(`["q","w","e"]`) as string[];
8   // const data = <string[]>JSON.parse(`["q","w","e"]`);
9
10  const array = [] as string[];
11  // const array = <string[]>[];
12  const num = 7 as number;
13  // const num = <number>7;
14
15
16  // TS allows coercion to a _more specific_ or
17  // to a _less specific_ type
18  type Gibberish =  'abc' | 'qwerty'
19  const str: Gibberish = 'qwerty'
20  const str2 = str as string
21  const str3 = 'whatever' as Gibberish
22
```

- [Playground link]

Type predicates

- Playground link

```
 9  type Animal = Fish | Bird;
10
11  function isFish(pet: Animal): pet is Fish {
12      return (pet as Fish).swim !== undefined;
13      // return 2;
14  }
15
16  const pet = {} as Animal;
17
18  if (isFish(pet)) {
19      pet.swim();
20  } else {
21      pet.fly();
22  }
23
```

Clicky-clicky time

- Types excersize

That’s it!

Kidding, homework!