title: Typescript 小技巧
date: 2019-08-19
tag: Typescript

description: 收集 Typescript 使用小技巧。

对象索引签名(Index signature)

```
const VERSIONS = {
  v1: 'V1',
  v2: 'V2',
}
```

使用 `typeof` 拿到类型定义

```
type VERSIONSType = typeof VERSIONS;
// 等价
type VERSIONSType = {
  v1: string
  v2: string
}
```

对象进行轮询操作：

```
Object.keys(VERSIONS).map(v => ({
  value: v,
  title: VERSIONS[v]
}))
// Throw Error
// => No index signature with a parameter
```

误区

定义一个 `indexSignature`

```
export interface IndexSignature {
  [key: string]: any;
}
```

然后添加类型访问

```
Object.keys(VERSIONS).map(v => ({
  value: v,
  title: (VERSIONS as IndexSignature)[v]
}))
```

索引将类型定义为 `any`，后导致后续无法进行类型推断检查，如果给对象每个 `key` 定义类型，右难以拓展维护（新 `key` 加入需要同步更改）

正解

`keyof` 拿到对象所有 `key` 的 `union` 类型。

```
type VERSIONSKey = keyof typeof VERSIONS;
(Object.keys(VERSIONS) as VERSIONSKey[]).map(v => ({
  value: v,
  title: VERSIONS[v]
}))
```

实例

```
() => {
  const VERSIONS = {
    v1: 'V1',
    v2: 'V2',
  }
  interface IndexSignature {
    [key: string]: any;
  }
  type VERSIONSKey = keyof typeof VERSIONS;
  return (
    <>
      <div>Bad</div>
      {Object.keys(VERSIONS).map(v => ({
        value: v,
        title: (VERSIONS as IndexSignature)[v]
      }))}
      <div>Good</div>
      (Object.keys(VERSIONS) as VERSIONSKey[]).map(v => ({
        value: v,
        title: VERSIONS[v]
      }))
    </>
  )
}
```

复合类型转换

Typescript 的复合类型分为两类，**set** 和 **map**。

- set 无序的、无重复元素的集合
- map 没有重复的键值对

```
// set
type Size = 'sm' | 'md' | 'lg';
// map
interface Ob {
  a: string
  b: number
}
```

转换

```
type ObKeys = keyof Ob; // 'a' | 'b'
type ObVals = Ob[keyof Ob] // string | number

type SizeMap = {
  [k in Size]: number
}
// 等价
type SizeMap = {
  sm: number
  md: number
  lg: number
}
```

常用工具

set 转 map `Record<Set, type>`

```ts
type Record<K extends keyof any, T> = { [P in K]: T };
type Size = 'sm' | 'md' | 'lg';
type SizeMap = Record<Size, number>
// type SizeMap = {
//   sm: number
//   md: number
//   lg: number
// }
```

选 map 部分 `Pick<Map, Size>`

```ts
type Pick<T, K extends keyof T> = { [P in K]: T[P] };
type PickSizeMap = Pick<SizeMap, 'sm' | 'md'>;
// type SizeMap = {
//   sm: number
//   md: number
// }
```