title: Typescript 类型
date: 2019-09-12
tag: Typescript

description: Typescript ES2015 类型

梳理 `Typescript` 的类型使用。

使用

Typescript ECMAScript APIs 的定义。

转换

Partial 全部指定为可选类型

示例

```
type Foo = {
    a: number;
    b: string;
    c: boolean;
};

type FooPartial = Partial<Foo>;
//=> { a?: number; b?: string; c?: boolean }
```

实现

```
type Partial<T> = {
  [P in keyof T]?: T[P];
};
```

部分指定为可选类型

实现

`Omit 去掉指定字段联合 Pick 出指定字段设置为可选`

```
type WithOptional<T, K extends keyof T> = Omit<T, K> & Partial<Pick<T, K>>;
```

示例

将 name、endpoint 标记为可选

```
type AccountFormType = {
  name: string,
  endpoint: string,
  accessKey: string,
  secretKey: string,
  region?: string,
}
type AccountFormOptionalType = WithOptional<AccountFormType, 'name' | 'endpoint'>
// =>
// {
```

```
//   name?: string;
//   accessKey?: string
//   secretKey: string
//   endpoint: string
//   region?: string
// }
```

过滤

Pick 选取指定的类型

示例

```
type Foo = {
    a: number;
    b: string;
    c: boolean;
};

type FooWithAC = Pick<Foo, 'a' | 'c'>;
//=> { a: number; c: boolean };
```

实现

```
type Pick<T, K extends keyof T> = {
  [P in K]: T[P];
};
```

Omit 去掉指定的类型。

示例

```
type Foo = {
    a: number;
    b: string;
    c: boolean;
};
type FooWithoutA = Omit<Foo, 'a' | 'c'>;
//=> { b: string };
```

实现代码

```
type Except<T, K extends keyof T> = Pick<T, Exclude<keyof T, K>>
```

Record

示例

```
type Foo = {
    a: number;
    b: string;
    c: boolean;
};
type FooRecordWithString = Record<string, Foo>
//=> { [x: string]: Fo }
```

实现

```
type Record<K extends keyof any, T> = {
    [P in K]: T;
}
```

Exclude 排除可分配的类型

示例

```
type Foo = {
    a: number;
    b: string;
    c: boolean;
};

type FooKey = keyof Foo; //=> "a" | "b" | "c"
type FooKeyWithoutAC = Exclude<KeyOf, 'a' | 'c'> //=> "b"
```

实现

```
type Exclude<T, U> = T extends U ? never : T;
```

Extract 选择可分配的类型

示例

```
type Foo = {
    a: number;
    b: string;
    c: boolean;
};

type FooKey = keyof Foo; //=> "a" | "b" | "c"
type FooKeyWithAC = Extract<KeyOf, 'a' | 'c'> //=> "a" | "c"
```

实现

```
type Extract<T, U> = T extends U ? T :  never;
```