[TypeScript] 04 Decoding Perplexing

BASIC	string, number, boolean, union
TYPE MANIPULATION	generics, conditionals, keys, infer, mapped types + modifiers, index access
UTILITY TYPES	exclude, extract□pick, omit
OBJECT TYPES	optional properties

Use generics, unions, conditionals

```
type IsTruthy<T> = T extends null | undefined | false | 0 | "" ? false : true;

// equals

type IsTruthy<Y> =
if: T extends null | undefined | false | 0 | ""
then: false
else: true;
```

Use utility types

utility types □□□TS□□□MappedTypes□□□□□□

- Exclude, Returns the provided union, minus any union items to exclude.
- Pick, Return a new type from the provided type with the keys.

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

type ValidDoubleObject<T> = T extends { a: infer U; b: infer U } ? U : never;

// IDE Loose Autocomplete□□□ □string & {}□; □□ string□ModelNames□□string□Autoco
type ModelNames = "gpt-4o" | "o3-mini" | "claude-sonnet-latest" | (string & {});
const model: ModelNames = "others"; // □□□□□□□□□ Autocomplete
```

Use mapped types

```
// -? means remove optional modifier(□□ Optional □□).
type MappedValidType<T> = {[K in keyof T]-?: true};
type ValidKeyUnion<T> = {[K in keyof T]-?: true}[keyof T];

type ValidKeys<T> = {
[K in keyof T]-?: (Exclude<T[K], null>) extends { a: infer U; b: infer U }
? IsTruthy<U> extends true ? K : never
: never
}[keyof T]

// □□ Complex Type□□□□ util □□□ type □□□□
type Prettify<T> = { [K in keyof T]: T[K]} & {};
```

Date: 2024-08-10
Words: 320
Time to read: 1 min

Newer Older

2024-08-22
React Tips

2024-07-21
Chrome Network □□□□