[TypeScript] 01 Type vs Interface

Type Basic

```
// basic syntax
type someType = somePrimitives | Object;
const val: someType = someValue;

// interface
interface Foo {
[x: any]: someType;
}

// foo-class.d.ts
/**
* .d.ts □□□□□□□□□□□□, □□ declare □□□□□□□□□□ export □□
* □□□□□□ export□□□□□□□□□□□□ ts □□□□□□□ ts □□□□□□□□□□□□□□□□□ import □□
*/
declare class FooClass {
someProp: someType;
someMethod(someArgs: someType): someType;
}
```

Type VS Interface

> type[1] □□□□□□□□□
> □□□□□□interface □□□□□□□statement□□□ type □□□□□□□□expression□
> interface□□□c □□struct□□□□□□□□object□□□

```
type FooBar = {
foo: Foo
bar: Bar
}

// □ base type
// □ □□interface□□□□□□□□□□□□□□□□□□□□Object Based□
type BaseType = string
interface IBase = string // □□□

interface IFooBar {
foo: Foo
bar: Bar
}

// □ union type□interface □□□
type UnionType = string | number

// □ □□□□□□□□ IFooBar□TS□□□□□□□□□□□□□;
// □ □□□Type□□□TS□□□□□□□□□
interface Window {
test: string
}
window.test = "interface bad feature"; // Bad

// □ combined type
type FooBarCombined = { foo: Foo } & { bar: Bar } & { fooBar: FooBar }
// equals
interface IFoo {
foo: Foo
}
interface IBar {
bar: Bar
}
interface IFoobarCombined extends IFoo,IBar {
fooBar: FooBar
}
```

Type Predicates

```
type Employee = { name: string, email: string }
type UserOrEmployee = { name: string } | Employee;

const people: UserOrEmployee[] = [
//...
];
people.forEach(person => {
if (isEmployee(person)) {
console.log(person.email); // □□□type predicates□□□□□
}
})

// function isEmployee(person: UserOrEmployee) {
function isEmployee(person: UserOrEmployee): person is Employee {
return "email" in person;
}
```

Advance

Example 1: narrow types

```
const INVALID_STATUS = ["deleted", "rejected"] as const;

type InvalidStatus = (typeof INVALID_STATUS)[number];
type ValidStatus = "pending" | "in_progress" | "done";
type Status = InvalidStatus | ValidStatus;

// Type Predicates
function isInvalidStatus(status: Status): status is InvalidStatus {
return INVALID_STATUS.includes(status);
}

function getTaskDesc(task: Task): string {
// □□□□□□□□□□
// if (task.status === "deleted" || task.status === "rejected")
if (isInvalidStatus(task.status) {
return `Task ${task.id} is Invalid!`;
}
return getValidTaskDesc(task.status);
}

function getValidTaskDesc(status: ValidStatus) {
// switch case expression
}
```

Example 2: TS check

```
type Status = ...;
// const TYPE_BY_STATUS: Record<string, string> = {
const TYPE_BY_STATUS: Record<Status, string> = {
deleted: "Deleted",
// rest
}
```

Exan

---

1. https://www.youtube.com/watch?v=oiFo2z8ILNo □

Date: 2024-02-26
Words: 579

Time to read: 2 mins

Newer Older

2024-02-26
[TypeScript] 02 Generic

2024-02-24
FAQ_001: CrossSite/□□