TypeScript

Type & Interfarce

```ts
interface Person ={
    name: string;
    age: number;
}

const john: Person = {
    name: "John",
    age: 26,
};
```

```ts
type Person = {
    name: string;
    age: number;
};

const john: Person = {
    name: "John",
    age: 26,
};
```

```ts
interface Person ={
    name: string;
    age: number;
}

const john: Person = {
    name: "John",
    age: 26,
};
```

≠

```ts
type Person = {
    name: string;
    age: number;
};

const john: Person = {
    name: "John",
    age: 26,
};
```

Interface

Members' declarations

↓

Properties and methods

```
interface Person ={
    name: string;
    age: number;
}

const john: Person = {
    name: "John",
    age: 26,
};
```

```ts
type Person = {
    name: string;
    age: number;
};

const john: Person = {
    name: "John",
    age: 26,
};
```

Type

Data Type

String, Boolean, number, other types (primitive, tuple, etc)

Declaration Merging

```ts
interface Person {
    name: string;
}

interface Person {
    age: number;
}

const john: Person = {
    name: "John",
    age: 26,
};
```

Declaration Merging

```ts
interface Person {
    name: string;
}

interface Person {
    age: number;
}

const john: Person = {
    name: "John",
    age: 26,
};
```

```ts
type Person = {
    name: string;
}

type Person = {
    age: number;
}
```

⊗ Duplicate identifier 'Person'.

Extends & Intersection

```
interface PersonNameInterface { name: string }
interface Person1 extends PersonNameInterface { age: number }
```

Inheritance of parent interface members.

Extends & Intersection

```ts
type PersonNameType = { name: string }
interface Person2 extends PersonNameType { age: number }

class PersonClass { name = "Jhon" }
interface Person3 extends PersonClass { age: number }
```

Extends & Intersection

```ts
type PersonNameType = { name: string; }
type Person1 = PersonNameType & { age: number; }

interface PersonNameInterface { name: string; }
type Person2 = PersonNameInterface & { age: number; }
```

Implements & Union

Distinct concepts ⟶ Definition and manipulation of types

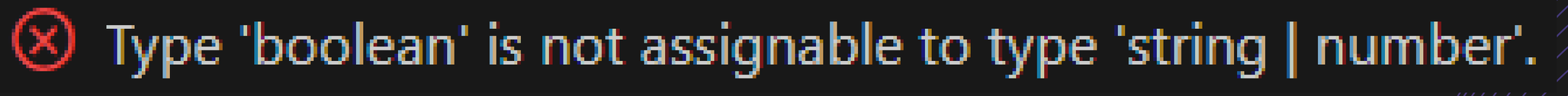
Implements & Union

```ts
interface IPerson {
  name: string;
  age: number;
}

class Person implements IPerson {
  constructor(public name: string, public age: number) {}
}
```

Implements & Union

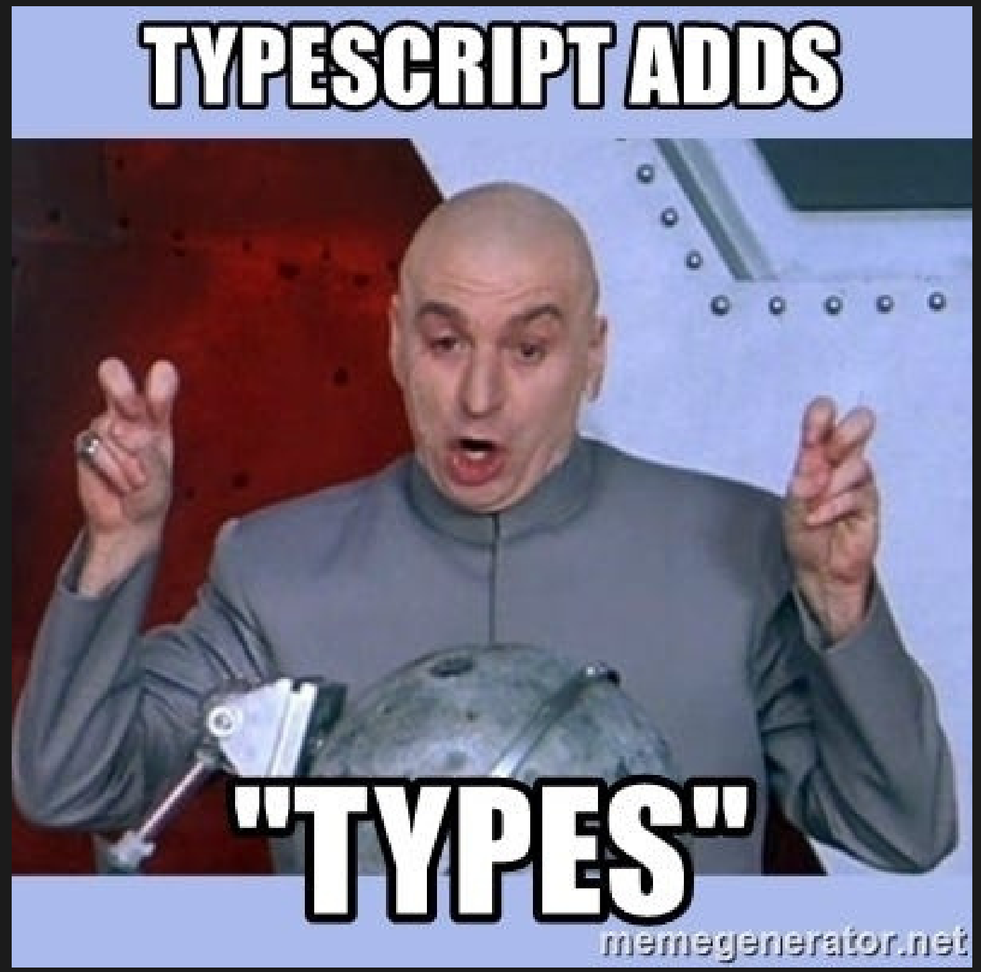
```ts
let value: string | number;
value = "Hello"; // OK
value = 42; // OK
value = true;
```



What should we use and when?

Conclusion

Thank You !



Source

- Type vs Interface in TypeScript