Read Only & Optional

```
26
27  type User = {
28      readonly _id: string
29      name: string
30      email: string
31      isActive: boolean
32      credcardDetails?: number
33  }
34
35  let myUser: User = {
36      _id: "1245",
37      name: "h",
38      email: "h@h.com",
39      isActive: false
40  }
41
42  myUser.name="mohit";
43  myUser._id="123";
44
```

while working with DB we dont want user to write id so we put readOnly on id

CreditCardDetails is optional to put? as Not Every user has a credit card

id is read Only so it shows Error

If we compile this, it will be compiled into a JS file but TypeScript will give Error Although you have error in TypeScript you will get the own JS file perfectly fine.

```
45  type cardNumber = {
46      cardnumber: string
47  }
48
49  type cardDate = {
50      cardDate: string
51  }
52
53  type cardDetails = cardNumber & cardDate & {
54      cvv: number
55  }
56  let  cd:cardDetails={
57      cardnumber:"abc",
58      cardDate:"def",
59      cvv:123
60  }
61
62  console.log(cd);
63
```

when we want to use other types Kind of Inheritance

After compiling the JS code

Pretty Simple

002.basics > JS Objects.js > ...

```
1  "use strict";
2  // const User = {
3  //     name: "hitesh",
4  //     email: "hitesh@lco.dev",
5  //     isAvtive: true
6  // }
7  Object.defineProperty(exports, "__esModule", { value: true });
8  var cd = {
9      cardnumber: "abc",
10     cardDate: "def",
11     cvv: 123
12 };
13 console.log(cd);
14
```

Arrays

some special built-in types that will very rarely appear in your code. *Always* use `string`, `number`, or `boolean` for types.

On this

Arrays

To specify the type of an array like `[1, 2, 3]`, you can use the syntax `number[]`; this syntax works for any type (e.g. `string[]` is an array of strings, and so on). You may also see this written as `Array<number>`, which means the same thing. We'll learn more about the syntax `T<U>` when we cover *generics*.

> Note that `[number]` is a different thing; refer to the section on Tuples.

any

myArray.ts > ...

```
1  const superHeros = []
2
3
4  superHeros.push("spiderman")
```

We have created a SuperHeros[] Array & pushed 'spiderman' to it Now see the error bellow

myArray.ts 1 •

myArray.ts > ...

```
1  const superHer
2
3
4  superHeros.push("spiderman")
```

Argument of type 'string' is not assignable to parameter of type 'never'. ts(2345)

View Problem No quick fixes available

Says String is not assignable to parameter type never

If you do Const SuperHeroes : [] = [] it still mean you haven't define the Array type there you have just defined that SuperHeroes is a type of Array

Now see Syntax of how to define Array type as String

```
myArray.ts > ...
1  const superHeros: string[] = []
2
3
4  superHeros.push("spiderman")
```

There is one more way to declare Array

Const SuperHeroes : Array <string> = []

put this in place of string[]

```
002.basics > TS arrays.ts > ...
1   const superHeros: string[] = []
2   // const heroPower: number[] = []
3   const heroPower: Array<number> = []
4
```

Two ways to declare Array

```
6
7   type User = {
8       name: string
9       isActive: boolean
10  }
11
12  const allUsers: User[] = []
13
14  allUsers.push({name: "", isActive: true})
15
```

Array of user

This push can't accept Empty objects

```ts
const MLModels: number[][] = [
    [255, 255, 255],
    []
]
```

2D Array Declaration of number

Union Type

Union Type is used when we do not know which type of data will come so there we specify Multiple types of data type In a case where you dont know about types dont use ANY Just try to use Union

```ts
let score: number | string = 33
score = 44
score = "55"
```

Type Just OR Symbol

It is advised to put type Strictly

We can even use union with user defined data types!!

```ts
type User = {
    name: string;
    id: number
}

type Admin = {
    username: string;
    id: number
}

let hitesh: User | Admin = {name: "hitesh", id: 334}

hitesh = {username: "hc", id: 334}
```

Perfect

hitesh as User

hitesh as Admin

```ts
function getDbId(id: number | string){
    //making some API calls
    console.log(`DB id is: ${id}`);

}
getDbId(3)
getDbId("3")

function getDbId1(id: number | string){
    if (typeof id === "string") {
        console.log("id is from fun 1 in string "+id.toLowerCase());
    }
  console.log("is is in 1 in f "+id);

}
getDbId1(3)
getDbId1("MOHIT")
```

```
[Running] node "c:\Users\mohit\Desktop\progr
DB id is: 3
DB id is: 3
is is in 1 in f 3
id is from fun 1 in string mohit
is is in 1 in f MOHIT
```

I guess no need to Explain Very Much Simple it is !!

Here if we do line 30 out of if then Typescript will show error as id can be no also so need to put inside if so that whenever we do .toLowerCase() it makes sure id is string

```ts
const data: number[] = [1, 2, 3]
const data2: string[] = ["1", "2", "3"]
const data3: (string | number | boolean)[] = ["1", "2", 3, true]

let seatAllotment: "aisle" | "middle" | "window"

seatAllotment = "aisle"
seatAllotment = "crew"
```

→ Array of mix of string, no & boolean

→ But we put crew so error

→ Seat allotment can be aisle, middle or window

```ts
// //array

const data: number[] = [1, 2, 3]
const data2: string[] = ["1", "2", "3"]
const data3: (string | number | boolean)[] = ["1", "2", 3, true]
Type '"crew"' is not assignable to type '"aisle" | "middle" | "window"'. ts(2322)

let seatAllotment: "aisle" | "middle" | "window"

View Problem (Alt+F8)  No quick fixes available
seatAllotment = "crew"
```

→ Array of no & string differently

See Error it got

Const data : string[] | number[] = {1, "2", 3}

It still gives Error

↓

This says Either Array of String or Array of Number

Instead use

const data : (string | number)[] = [1, "2"]

Dont use Any if there are Union of Multiple type as it defeat whole purpose of TypeScript Use Union & specify Type

```
2 , 3, true]
40
41 let pi:3.14 = 3.14
42 pi = 3.145
```

} when we want constants we put Type as Value

Suppose Sex can take 3 values "Male", "Female", "Trans"

let Sex: "M" | "F" | "T"

Tuples → you might have seen in Python But it is not well implemented here

We want a Array of string, number & then boolean in this order only

Tuple Types

A *tuple type* is another sort of `Array` type that knows exactly how many elements it contains, and exactly which types it contains at specific positions.

```
type StringNumberPair = [string, number];
```

Here, `StringNumberPair` is a tuple type of `string` and `number`. Like `ReadonlyArray`, it has no representation at runtime, but is significant to TypeScript. To the type system, `StringNumberPair` describes arrays whose `0` index contains a `string` and whose `1` index contains a `number`.

Tuple is fixed size Array & type of each position is defined !!

```ts
function doSomething(pair: [string, number]) {
  const a = pair[0];
        const a: string
  const b = pair[1];
        const b: number
  // ...
}

doSomething(["hello", 42]);
```

from documentation tuple straight away. Corn

If we try to index past the number of elements, we'll get an error.

```ts
function doSomething(pair: [string, number]) {
  // ...

  const c = pair[2];
Tuple type '[string, number]' of length '2' has no element at index '2'.
}
```

```ts
let rgb: [number, number, number] = [255, 123, 112,0.5]
```

See error as we have defined tuple to be 3 & of number type all 3

```ts
type User = [number, string]

const newUser: User = [112, "example@google.com"]

newUser[1] = "hc.com"
//newUser.push(true)
```

You can change values later But with same type

You will not be able to push true as it says Array only accept String | number type

```ts

type User = [number, string]

const newUser: User = [112, "example@google.com"]

newUser[1] = "hc.com"
newUser.push("mohit")
console.log(newUser);

```

→ It works

```
[Running] node "c:\Users\mohit\Desktop\programs\typescript\002.basics\tuple.js"
[ 112, 'hc.com', 'mohit' ]
```

→ Works

But it should not be working as size of Array needs to be fixed So its a problem