Interface

It looks similar to TYPE but it's NOT.

Let's take an example.

According to the manual interface are faster as compared to types

function eg1(){

type Student = {

rollNumber:number;

name:string;

}

interface Employee{

id:string;

name:string;

}

let a:Student;

let b:Employee;

a = {

rollNumber:101,

name:"Ramesh"

};

b={

id:"A1001",

name:"Suresh"

};

console.log(a);

console.log(b);

}

eg1();

/\*

{ rollNumber: 101, name: 'Ramesh' }

{ id: 'A1001', name: 'Suresh' }

\*/

.

We can use variable of interface with 2 ways

function eg3(){

interface NewsPaper{

headline:string;

}

let k:NewsPaper;

k = {

headline:"God is great"

}

console.log(k);

k.headline = "Ujjain is the city of god";

console.log(k);

}

eg3();

If we define a variable as read-only, we can not assign value to it.

function eg4(){

interface NewsPaper{

readonly headline:string;

}

let k:NewsPaper;

k = {

headline:"God is great"

}

console.log(k);

k.headline = "Ujjain is the city of god"; //Error- Can not assign

console.log(k);

}

eg4();

interface NewsPaper{

readonly headline:string;

}

function eg5\_some(np:NewsPaper){

console.log(np);

np.headline = "Cool place";

console.log(np);

}

Here we define parameter of type NewsPaper and that have readonly property.

So its not allowing us to use the code even we not yet write code to call it.

Because when we have created the object we had defined the headline. So its wrong.

Let's see

interface NewsPaper{

readonly headline: string;

}

function eg5\_some(np:NewsPaper){

console.log(np);

np.headline = "Cool place";

console.log(np);

}

function eg5(){

let k:NewsPaper;

k = {

headline: "Good";

}

}

eg5();

Important example

We are passing data of type NewsPaper to a function thats type not defined

// important

interface NewsPaper{

readonly headline:string;

}

function eg8\_some(np){ // but here its inter to type ANY . so it will work

console.log(np);

np.headline = "Cool place";

console.log(np);

}

function eg8(){

let m = {

headline: "Ujjain is the city of gods"

}

eg8\_some(m);

let k:NewsPaper

k = {

headline: "God is great"

}

eg8\_some(k); // we are passing input of type NewsPaper

}

eg8();

/\*

{ headline: 'Ujjain is the city of gods' }

{ headline: 'Cool place' }

{ headline: 'God is great' }

{ headline: 'Cool place' }

\*/

Its working because its type is considered as any

So when defining any function, we will determine its parameter type.

Proof we are not creating call its function

See code

function eg9(){

interface Bulb{

wattage:number,

printWattage():void

}

let b:Bulb;

b = {

wattage:60,

printWattage():void{

console.log(this.wattage);

}

}

console.log(b);

let c:Bulb;

c = {

wattage:240,

printWattage():void{

console.log(`wattage is ${this.wattage}`);

}

}

console.log(c);

}

eg9();

/\* output:

{ wattage: 60, printWattage: [Function: printWattage] }

{ wattage: 240, printWattage: [Function: printWattage] }

\*/