React Native app

To create application we have 2 options

JavaScript

* Js is very powerful but its Not Strongly Typed language
* Nothing Wrong with Js but if we have Average programmers then its hard to find bugs and difficulty in development due to its Dynamic nature.

TypeScript (Someone from Microsoft make this)

* Its SuperSet of JS.
* Compiler (Transpiler) (TSC)
* Its programming Language which underline technology is Javascript.
* Means Typescript code will not be executed directly first it will convert into js code then js code will compile.

When Transpiler will show error but still it will create JS File

So its developer responsibility to solve ts bugs.

Install Node (it includes npm)

Install npx (npm i -g npx)

Install tsc (npm i -g typescript)

Write first program in ts.

> Vim eg1.ts

// write code and save

> tsc eg1.ts // this will convert ts file into JS File and save.

> node eg1.js

// output will execute.

Now we want we will write all ts code in 1 file and it will compile all code at once at another place. So we have to change some values in configuration.

> tsc –init

Created a new esconfig.json with

Target: es2016

Module commonks

Strict: true

esModleInterop: true

skipLibCheck:true

// it will create tsconfig.json with above configuration

// update target from es2016 to es2022

// and we want js will will transpile and created in js Directory.

// for that we will output Directly path.

"target": "es2016", => "target": "es2022",

// "outDir": "./", => "outDir": "./js",

Mkdir js

ts> vim eg2.js

function eg2()

{

let x = 10; // data type of x will be implicitly set to number.

console.log(typeof x);

let y = null; // data type of y will be implicitly set to null

console.log(typeof y);

let z:null; // data type of z will be explicitly to null type

z=null;

console.log(typeof z);

}

eg2();

ts>tsc

ts> node js/eg2.js

>

number

object

object

// here we can see we have given data type number to x and null to y & z

But when we execute js file we can see its data type showing is Object.

Here we can see clearly. We are typing code in ts but still underline code is written in javascript and execution is also based on js only.

When we compile code then it we are getting error

We have to fix them

Otherwise there is no sense of using type script if we are getting profile still we are contineulsy working on it without fixing it because the file of js is getting generated.

Make a simple rule

**NOTE: If we are getting error in ts file then assume we are JS file is not generated and fix and again compile**

function eg4()

{

let a:string|number;

a = "Good";

console.log(a);

if(typeof a == "string")

console.log("Date type of a is string");

else

console.log("Date type of a is not string");

a = 20;

console.log(a);

if(typeof a == "number")

console.log("Date type of a is number");

else

console.log("Date type of a is not number");

a = "Great";

console.log(a);

if(typeof a == "string")

console.log("Date type of a is string");

else

console.log("Date type of a is not string");

let b = null;

if(typeof b == "null")

console.log("Date type of a is NULL");

else

console.log("Date type of a is not NULL");

}

eg4();

/\*

eg4.ts:25:4 - error TS2367: This comparison appears to be unintentional because the types '"string" | "number" | "bigint" | "boolean" | "symbol" | "undefined" | "object" | "function"' and '"null"' have no overlap.

25 if(typeof b == "null")

~~~~~~~~~~~~~~~~~~

Found 1 error in eg4.ts:25

-----------------------------

here we are definfing data type as per ts

but its comparing as par javascript

means if you are comparing the type of any data

then you should use data types of javascript not ts

becase when we give type null in ts, we got type object in js

\*/

function eg10()

{

type student={

firstName:string,

// lastName:string,

lastName?:string, // here we are giving optional value

}

let a:student={

firstName:"Rahul"

}

}

eg10();

/\*

lastName:string,

due to this line we will get error

but when we put ?

then it will take value as optional

lastName?:string,

\*/

function eg12()

{

let a ={

wattage:60,

brand:"Philips",

}

console.log("price" in a);

console.log("wattage" in a);

console.log("brand" in a);

}

We can determine variable is available in json or not by above code. (In)

type Student={

name:string,

city:string,

dataType?:"Student"

};

let s:Student;

s = {

name:"Ramesh",

city:"Ujjain",

dataType:"Student"

};

console.log(s);

console.log(typeof s);

/\*

if(typeof s == "Student")

console.log("S is of type Student");

else

console.log("S is not of type Student");

// eg13.ts:16:5 - error TS2367: This comparison appears to be unintentional because the types '"string" | "number" | "bigint" | "boolean" | "symbol" | "undefined" | "object" | "function"' and '"Student"' have no overlap.

//

// 16 if(typeof s == "Student")

// ~~~~~~~~~~~~~~~~~~~~~

//

//

// Found 1 error in eg13.ts:16

\*/

if(s.dataType == 'Student')

console.log("S is of type Student");

else

console.log("S is not of type Student");

—

Union of two types

type Cake = {

name: string,

price: number,

};

type IceCream = {

price: number,

flavour: string,

};

type iceCreamCake = Cake & IceCream;

let i:iceCreamCake;

i = {

name: "Cool Cake",

price:200,

flavour:"Vanilla"

}