بررسی نکات تایپ اسکریپت

جلسه اول

1- خب قاعدتا اول محیط رو آماده میکنیم اول node رو نصب کرده

2- npm install -g typescript

3- محیط توسعه رو باز میکنیم با ide و سپس با دستور npm init اماده میشیم که بتونیم پکیج هایی که میخواهیم رو نصب کنیم

4- tsc app.ts🡪app.js

5- وقتی initمیکنیم npm رو تو پکیج جیسون برای ران کردن یه سری دستورات میتونیم اسکریپت بنویسیم که خیلی کمکمون میکنه

جلسه دوم

1. خوبی ts اینه که تو فرایند دولوپ چک میکنه تایپ رو ولی js در زمان اجرا یا ران تایم (کنسول مرورگر)
2. منظور از هندل کرد تایم توسط js این بود که مثلا یه شرطی چیزی بنویسیم اینطوری :

function add(num1: number, num2: number): any {  
 if (typeof num1 != "number" && typeof num2 != "number")  
 {  
 throw new Error('invalid type ...!!!!!!!')  
 }  
 return num1 + num2  
}

console :

app.js:5 Uncaught Error: invalid type ...!!!!!!!

at add (app.js:5:15)

at app.js:11:11

پس با ts تو محیط دولوپ و ترمینال دولوپ میتونیم بفهمیم چه باید کرد پس حواسمون باشه!

\*\*\*\*\*\*

1. حواسمون باشه وقتی آبجکتی تعریف میکنیم حتما نوعش رو مشخص کنیم که امکان خطا بیاد پایین :
2. const person: {  
    name: string,  
    age: number  
   } = {  
    name: 'saeed',  
    age: 28  
   }  
     
   console.log(person.name)  
     
   //output : saeed

4- آرایه

const product = {  
 name: 'code12',  
 conutry: 'us',  
 cur: ['usdt', 'usd', 'ir', 'eur']  
}  
  
for (let item of product.cur) {  
 console.log(item.toUpperCase())  
}  
  
  
const currency: string[] = ['usdt', 'usd', 'ir', 'eur']  
console.log(currency)

5-tuple : در آرایه تمام تایپ هاشو مشخص میکنیم

var employee: [number, string] = [1, "Steve"];

var person: [number, string, boolean] = [1, "Steve", true];

var user: [number, string, boolean, number, string];// declare tuple variable

user = [1, "Steve", true, 20, "Admin"];// initialize tuple variable

1. enum :

An enum is a special "class" that represents a group of constants (unchangeable variables).

Enums come in two flavors string and numeric.

enum Color {  
 BLUE,  
 RED,  
 GREEN  
}  
  
const product = {  
 name:'sony',  
 price:120,  
 color:Color.BLUE  
}  
if (product.color===Color.BLUE)  
{  
 console.log('product doesnt exist')

جلسه سوم

1. تمام تلاشون باید این باشه که از تایپ any استفاده نکنیم

یعنی در واقع بهش میگیم نوع داده ای رو چک نکن

let product: any;  
product = [2, 56, 'ali']  
  
let product2: any[]  
product2 = 5;  
//error becouse we limited any to array

1. Union type : خب زمانی که ما نمیدونیم اون داده ای که داریم دقیقا چی هستش از یونیون تایپ استفاده میکنیم (در واقع معنی همون"یا" رو میده با علامت | ، به کد زیر دقت کن تا بفهمی !)

function sum(input1: number | string, input2: number | string) {  
 return input1 + input2;  
}  
  
console.log(sum(5, 10));  
console.log(sum('saeed', 'dashti'))

تو این حالت وقتی ران میکنم اپلیکشین رو تو کنسول خروجی که میخوایم رو به دست اوردیم

ولی همچنان تو ide یه ارور داریم که اینطوری با تغییر سینتکس میتونیم هندلش کنیم :

let res: number | string;  
  
function sum(input1: number | string, input2: number | string) {  
 if (typeof input1 === 'number' && typeof input2 === 'number') {  
 res = input1 + input2;  
 } else  
 res = input1.toString() + input2.toString()  
 return res;  
}  
  
console.log(sum(5, 10));  
console.log(sum('saeed', 'dashti'))

1. Aliases : ما میتونیم یه سری تایپ دیلخواه تعریف کنیم   
   و مقداری دهی کنیم و هر جا خواستیم ازشون استفاده کنیم

واقعیتش اینا خیلی نوشتنی نیست کد و ببین !

//declare type  
type Combine = number | string  
type CombineType = 'is-num' | 'is-txt'  
  
  
let res: Combine;  
  
function sum(input1: Combine, input2: Combine, resType: CombineType) {  
 if ((typeof input1 === 'number' && typeof input2 === 'number') || resType === 'is-num') {  
 res = +input1 + +input2;  
 } else  
 res = input1.toString() + input2.toString()  
 return res;  
}

1. تایپ void به توابعی داده میشود که مقدار بازگشتی ندارد (return ندارد)

function sum(num1: number, num2: number): number {  
 return num1 + num2;  
}  
  
function log(num: number):void {  
 console.log('result :' + num);  
}  
  
log(sum(2,7));  
  
//result :9

جلسه چهارم

1. ما یه تایپ فانکشن هم داریم که میتونیم بگیم که تایچ ما فانکه و ارگومان ها و همینطور مقدار بازگشتیشپ

چه تایپی باشه به کد زیر دقت کن :

function sum(num1: number, num2: number): number {  
 return num1 + num2;  
}  
  
function log(num: number): void {  
 console.log('result :' + num);  
}  
  
let result: (a:number,b:number) => number;  
result = sum  
console.log(result(6, 10))

\*\*\*اگر مقدار بازگشتی نداشت جلوی arrow عبارت void را میگذاریم

1. برای این که هربار کد رو کامپایل نکنیم میتونیم از فلگ watch استفاده کنیم

tsc app.ts –watch

tsc app.ts -w

1. خب ما ممکنه چنتا فایل ts داشته باشیم که باید کامپایل بشن اینطوری نمیتونیم دونه دونه هندل کنیم کار رو پس در مسیر روت پروژه مون دستور tsc –init رو میزنیم که یه فایل کانفیگ ts برامون ساخته میشه

بعدش وقتی تو ترمینال tsc -w رو بزنیم تمام فایل های tsمون برامون کامپایل میشه (باقی تنظیمات ts config هم جلوتر خواهیم دید ولی بهتره سرچ کنی )

1. rootDir مسیری هست که میتونیم فایل های ts رو قرار بدیم توش

outDir مسیری هست که فایل های js بعد از کامپایل توش قرار میگیره!

1. مفهوم کلاس در ts

class Team {  
 team: string  
  
 constructor(n: string) {  
 this.team = n;  
 }  
}  
  
const newTeam=new Team('developer')  
console.log(newTeam)

1. یکی از کاربرد های مهم this در ts دستیابی به property هایی هست که در یک کلاس تعریف شده است