TypeScript and ES6 Summery   
---------------------------------

عشان بس نبقى عارفين ان ES6 , Typescript لسه متمش اعتمادها فى بروزارز كتير و لكن هى المستقبل ان شاء الله فالى هنعمله اننا هنتعلم ES6 و فى أداه جاهزه كده بتحول الكود من ES6 ل ES5 عشان يتقرأ و ده اسمه Transpiler  
أول حاجه احنا بنحمل أداه كل الى بتعمله انها تحول الكود ذى ما قولنا و دى اسمها Babel  
و لينكه اهه :D

<http://babeljs.io/>

الاداه دى مش بتشتغل الا لما تكون محمل ال NodeJS و بالتالى هتحتاج تحمله و هتحمله من هنا

<https://nodejs.org/en/>

هتحمل الريكومنديشن فيرشن بقه و هتسطبه و هيبقى لسه فاضل بقه جزء الترمينال سواء للبابل او للنود  
و هتسطب بقه   
التسطيب بياخد وقت كبير فأنا هريح دماغى و احول الكود على موقع بايل جى اس و خلاص

اهم حاجه فى التريمينال عشان تسطب التايب سكريبت اكتب ده و رنه

npm i -g typescript

1) NodeJS: [https://nodejs.org/en/](https://www.youtube.com/redirect?v=WmeiroTUQyo&event=video_description&redir_token=SRyBkoAQYmd9gVt9CKA5pbW283B8MTUzMjU0NTE1NkAxNTMyNDU4NzU2&q=https%3A%2F%2Fnodejs.org%2Fen%2F)

2) Visual Studio Code: [https://code.visualstudio.com/Download](https://www.youtube.com/redirect?v=WmeiroTUQyo&event=video_description&redir_token=SRyBkoAQYmd9gVt9CKA5pbW283B8MTUzMjU0NTE1NkAxNTMyNDU4NzU2&q=https%3A%2F%2Fcode.visualstudio.com%2FDownload) For further reading

on tsconfig.json [https://www.typescriptlang.org/docs/h...](https://www.youtube.com/redirect?v=WmeiroTUQyo&event=video_description&redir_token=SRyBkoAQYmd9gVt9CKA5pbW283B8MTUzMjU0NTE1NkAxNTMyNDU4NzU2&q=https%3A%2F%2Fwww.typescriptlang.org%2Fdocs%2Fhandbook%2Ftsconfig-json.html)

Link to the basic project already setup [https://github.com/amitauras/typescri...](https://www.youtube.com/redirect?v=WmeiroTUQyo&event=video_description&redir_token=SRyBkoAQYmd9gVt9CKA5pbW283B8MTUzMjU0NTE1NkAxNTMyNDU4NzU2&q=https%3A%2F%2Fgithub.com%2Famitauras%2Ftypescript-setup)

هتفهم من هنا احسن

<https://www.youtube.com/watch?v=WmeiroTUQyo>

اى ملف Typescript بتعمله لازم تحوله لجافا سكريبت عشان البراوزر يعرف يقراه و ده عن طريق الكوماند ده

**tsc -w FileName.ts او tsc -w و دى جلوبال بقه لكله**

و عشان تعمل كونفيجريشن بقه للتايب سكريبت بتكتب الكوماند ده و تخش تعدل على الكونفيجريشن لو عاوز

**tsc –init**

أول ما هتنفذ الكوماند ده هتروح عملالك فايل ال tsConfig.json و ده الخاص بالكونفيجريشن بتاعت التايب سكريبت كده خلاص خش ظبط فيه براحتك ياباشا

#######################################################################  
Typescript Summery

----------------------------  
let , var , const

الفرق بين التلاته دول ايه بقه   
بص يا سيدى ال var دى ملهاش سكوب جوه IF هى معتبره اقواس ال if كانها هواء هههههه و تناديها بره ال IF و تلاقيها شغاله عادى خالص & تانى حاجه ملهاش نوع معين يعنى بتعرف الحاجه و تعملها انشياليز ب سترينج او نامبر او اراى او اى حاجه

/\*eslint-env browser\*/

var Myname = "Mahmoud";

var Myage = 27;

var Myfamily = ["mahmoud", "Mohamed", "Mostafa", "karim", 25,

(function () {

'use strict';

return 5;

}())

];

window.console.log(Myfamily);

window.console.log(Myage);

window.console.log(Myname);

او

if(true)

{

var x = 5;

}

console.log(x);

ال let ليها سكوب ذيها ذى اى فاريبول فى اى لغه تانيه شايف عامل ايرور اذاى :D

if(true)

{

let MyNum = 5;

}

console.log(MyNum);

ذى ما انت شايف كده ال var مفيهاش حاجه اسمها تحديد نوع على عكس let

دى بقه بتتعامل مع كله عادى و كذلك اقدر برضوا احددلها نوع معين احبره يتعامل معاه   
فهتلاقىه بيتعامل مع الفاريبول كسترينج و كانمبر و كبوليان و كأراى و ادى مثال اهه

var x = [5,6,4] ;

var z = "Mostafa"

let Name: string="Mahmoud"; // accept String Only

let Num: number = 5;

let IsGood: boolean = true;

let MyArr1: string[]; // first Method to write Array of string

let MyArr2: Array<string>; // second Method (Generic) to write Array of string

let MyArr3: number[]; // first Method to write Array of number

let MyArr4: Array<number>; // second Method to write Array of number

let MyArrAny: Array<any>; // this array like list because now it accept any data type

عندك بقه حاجه اسمها Const دى بتثبت رقم بشكل جلوبال بحيث محدش يعرف يغيره و ده بيبقى معمول مثلا للكونيكشن سترينج

const Myname:string = "mahmoud";

Mebo

#######################################################################  
Type Assertion

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

دى بقه يا سيدى بتبقى معموله كده عشان انا ممكن استخدم الداتا تايب الى اسمها any و بعد كده احتاج انى استخدم فانكشنز معينه لداتا تايب بالتحديد ساعتها هو بيبقاش عارف الفاريبول دى تبع انهى داتا تايب فبيظهرش الميثودز بتعتها خالص   
طيب عشان احل المشكله دى بعمل ايه بص يا سيدى بتروح بقه محول الداتا تايب دى لاسترينج مثلا عشان تعرف تستخدم ميثودز الاسترينج طب ده يحصل اذاى . اهه ياعمنا

let Myname1:any ;

(<string>Myname1).concat("Hello");

//or

let Myname2:any ;

(Myname2 as string).concat("Hello")

//or

let Myname3:any = "Mahmoud" ;

Myname3.concat("Hello");

Mebo

#######################################################################  
Arrow Functions

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

الفانكشن العاديه بتبقى عامله كده

let Func = function MeboFunc(num) {

console.log(num + 5)

}

او

let Func = function (num){

console.log(num + 5)

}

او

let Func = function(Name:string):string {

return Name;

}

Func("Mahmoud Badawy")

او

function MeboFunc(Name:string):string {

return Name;

}

MeboFunc("Mahmoud Badawy")

و دى معناها انه بيقبل فقط استرينج و بيرجع استرينج

و بالتالى كده متوفر حته انى احدد النوع الى داخل و الى خارج من الفانكشن كده فى الجافا سكريبت دلوقتى بدل ما كانت سداح مداح

الفانكشن الى معموله بالارو بقه بتبقى كده arrow function

**var** *Func\_Original* = **function** (num) { ***console***.log(num + 5) };  
*Func\_Original*(5);  
  
-------------------------------------------------------------------------

**let** *Func* = (num) => { ***console***.log(num + 5) };  
*Func*(5);

و من مميزاتها بقه انها سهله فى الكتابه ذى ما انت شايف و بيقاش ليها اسم دايما عشان تبقى واخد بالك هى بترجع فى فاريبول ذى ما انت شايف كده و الفاريبول ده كمان الى بعمل بيه الكول و بترجع void الى هو يعنى ذى المثال الى فوق ده و ممكن احط كلمه فويد و ممكن لا و كذلك بترجع اى نوع من انواع الداتا تايب و بتقبل اى نوع من انواع الداتا تايب تعالى اوريك مثال حلو كده

**let** *MyArrowFunc* = (Name:**string** ,Age:**number** ,Friends:**string**[],  
 IsGoog:**boolean**):**string** =>{  
 **return `My Name is** ${Name} **, My Age is** ${Age} **,  
My First friend is** ${Friends[0]} **, He is** ${Friends[0]} **Man .`**};  
**let *Friend***:**string**[] = [**"Karim Salah"**,**"Msalah"**,**"Khaleed"**];  
***Friend***[3] = **"Tata"**;  
*MyArrowFunc*(**"Mahmoud Badawy"** , 27 , ***Friend***,**true**)  
  
---------------------------------------------------------------------------

بعد التحويل

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

**var** *MyArrowFunc* = **function** (Name, Age, Friends, IsGoog) {  
 **return "My Name is "** + Name + **" , My Age is "** + Age + **" ,\nMy First friend is "** + Friends[0] + **" , He is "** + Friends[0] + **" Man ."**;  
};  
**var *Friend*** = [**"Karim Salah"**, **"Msalah"**, **"Khaleed"**];  
***Friend***[3] = **"Tata"**;  
*MyArrowFunc*(**"Mahmoud Badawy"**, 27, ***Friend***, **true**);

الفانكشن دى بيدخلها مجموعه من الارجيومنت كل واحده بنوع ما و بترجع سترينجايه فى ريتيرن و من ضمن الحاجات الحلوه اوى فى الجزئيه الرتيرن حته ان فى سترينج فورمات كمان فى الجافا سكريبت و هو انى بحط الارجيومنت فى نص الاسترينج بكل سهوله فقط بحطه فى قوس مجموعه مسبوق بشكل الدولار ذى ده ${Name}و اهه ده ريحنا من موضوع الكونكاتيونيشن حتى كمان النيو لاين ده بيتعمل من نفسه هههههههه مش لازم اكتب سلاش ان :D \n بس اهم حاجه فى الاسترينج تيمبليت او الاسترينج فورمات دى انك تحطها وسط ` ` Single quotation البتاعه دى و دى موجوده عند حرف ال ( ذ )

Mebo

#######################################################################  
interface

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

دى بتنظم و تظبط الكود و هنا من الاخر بعمل انترفيس وبعمل منه اوبجيكت اجمع فيه الفاريبول و مش بيقبل فانكشنز عشان تبقى واخد بالك هو بينظم الفاريبولز بس بستخدمهم بقه فى اماكن تانيه و بدل ما استخدم الارجيومنتس 200 ارجيومنت فى الفانكشن بروح حاططهم فى انترفيس و اعمل ارجيبومنت من نوع الانترفيس ده ذى ما انت شايف تحت الكود سهل مش محتاج شرح كانك مستخدم list بالظبط بس ده بقه شكله كانك مستخدم كلاس بس مبيقبلش فانكشنز فاهمنى ؟؟

**interface** Person {  
 **Name** : **string** ,  
 **Age** : Number ,  
 **Height**? : Number *// ? Mean that is Optional*}  
  
**let *obj***:Person = {**Name** : **"Mahmoud"**, **Age** : 27 , **Height** : 173 };  
**let *obj2***:Person = {**Name** : **"Mahmoud"**, **Age** : 27 } ;  
  
**let** *getMyInfo* = (MainObj:Person):**string** => {  
 **return `Hi I'm** ${MainObj.**Name**} **, I'm** ${MainObj.**Age**} **My Height is** ${MainObj.**Height**}**`**;  
};  
***console***.log(*getMyInfo*(***obj***)) ;  
***console***.log(*getMyInfo*(***obj2***)) ;  
***console***.log(*getMyInfo*(  
 {**Name** : **"Mostafa"**, **Age** : 1 , **Height** : 40} *// on the fly object*)) ;  
  
**let *obj3***:Person;  
***obj3***.**Age** = 5 ;  
***obj3***.**Name** = **"Yasser"** ;  
***obj3***.**Height** = 100 ;  
***console***.log(*getMyInfo*(***obj3***)) ;

Mebo

#######################################################################  
OOP Class / Property / Constructor / Methods / Object / this

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

مختلفش خالص عن اى لغه عمتا بعمل كلاس و اديله اسم و جواه بحط بروبرتيس و كونستراكتور عشان اعمل اكسيس على الكلاس بالاوبجيكت بيه و طبعا ميثودز بتعمل حاجات معينه و طبعا الاوبجيكت عشان انادى بيها اى حاجه موجوده فى الكلاس من بره و this عشان انادى اى حاجه موجوده فى الكلاس من جوه الكلاس لانها بتعبر عن الاوبجيكت و لكن بداخل الكلاس و ادى كود و عمتا لو فاهم اى لغه عقيمه هتفهم الى مكتوب هنا ايه

**class** Car {  
 **Name**:**string** ; *// property*  
 **Model**:**string** ;  
 **Speed**:**number**;  
  
 **constructor**(CarName:**string** ,CarModel:**string** ,CarSpeed:**number** )  
 {  
 **this**.**Name** = CarName;  
 **this**.**Model** = CarModel;  
 **this**.**Speed** = CarSpeed;  
 ***console***.log(**'Hello From Constructor '**);  
 }  
  
*GetCarInfo* = (CarName:**string** ,CarModel:**string** ,CarSpeed:**number**):**string** =>{  
**return `Car Name :** ${CarName} **, Car Model :** ${CarModel} **, Car Speed :** ${CarSpeed}**`**;  
 }  
  
 *GetCarInfoByPassingFromConstractor* = ():**string** =>{  
 **return `Car Name :** ${**this**.**Name**} **, Car Model :** ${**this**.**Model**} **, Car Speed :** ${**this**.**Speed**}**`**;  
 }  
}  
  
**let *MyObj***= **new** Car(**'BadawyCar'**,**'Model 2018'** ,220); *// Create An object from Car****console***.log(***MyObj***.*GetCarInfo*(**'MeboCar'**,**'Model 2020'** ,5000));

*// Call the Function From the Class by Obj of Car****console***.log(***MyObj***.*GetCarInfoByPassingFromConstractor*());

#######################################################################  
inheritance الوراثه

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

بص ياعم دى عباره عن وراثه الابن للاب فى كل خصائصه العامه و البروتيكتيد

يعنى انا اب و ابنى ورث منى فبيورث كل الخصائص و كل الوظايف و كل حاجه ما عدا الحاجات الى انا عاملها خاصه بيا يعنى مثلا انا ياعم حالق شعرى هو ملوش دعوه مش هيورثها لان انا عامل حلاقه الشعر حاجه خاصه بيا بروح احلق شعرى كل شويه قرعه مثلا هو بيروح يحلقها سبايكى بقه ههههههههههه

مثال على الوراثه ده بقه يعنى هتلاقيه شبه اى لغه هبله برضوا

**class** Car {  
 **Name**:**string** ;  
 **Model**:**string** ;  
 **Speed**:**number**;  
  
 **constructor**(CarName:**string** ,CarModel:**string** ,CarSpeed:**number** )  
 {  
 **this**.**Name** = CarName;  
 **this**.**Model** = CarModel;  
 **this**.**Speed** = CarSpeed;  
 ***console***.log(**'Hello From Constructor '**);

}

*GetCarInfo* = (CarName:**string** ,CarModel:**string** ,CarSpeed:**number**):**string** =>{  
 **return `Car Name :** ${CarName} **, Car Model :** ${CarModel} **, Car Speed :** ${CarSpeed}**`**;  
 }  
  
 *GetCarInfoByPassingFromConstractor* = ():**string** =>{  
 **return `Car Name :** ${**this**.**Name**} **, Car Model :** ${**this**.**Model**} **, Car Speed :** ${**this**.**Speed**}**`**;  
 }  
}  
  
  
**class** ChildCar **extends** Car{  
  
}  
  
**let *MyObj***= **new** ChildCar(**'BadawyCar'**,**'Model 2018'** ,220); *// Create An object from* ChildCar***console***.log(***MyObj***.*GetCarInfo*(**'MeboCar'**,**'Model 2020'** ,5000)); *// Call the Function From the Class by Obj of* ChildCar***console***.log(***MyObj***.*GetCarInfoByPassingFromConstractor*());

واخد بالك ان الابن ورث من الاب كل خصائصه عادى خالص و كانه هو بالظبط لان الاب عامل كل حاجه public

#######################################################################  
overloading

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

ان الابن احط فيه فانكشن بنفس اسم الفانكشن الى فى الاب بس مع اختلاف البراميتر الى دخلاله

**class** Car {  
 **Name**:**string** ;  
 **Model**:**string** ;  
 **Speed**:**number**;  
  
 **constructor**(CarName:**string** ,CarModel:**string** ,CarSpeed:**number** )  
 {  
 **this**.**Name** = CarName;  
 **this**.**Model** = CarModel;  
 **this**.**Speed** = CarSpeed;  
 ***console***.log(**'Hello From Constructor '**);  
 }  
  
*GetCarInfo* = (CarName:**string** ,CarModel:**string** ,CarSpeed:**number**):**string** =>{  
 **return `Car Name :** ${CarName} **, Car Model :** ${CarModel} **, Car Speed :** ${CarSpeed}**`**;  
 }  
  
}  
  
**class** ChildCar **extends** Car{  
 *//OverLoading Function  
 GetCarInfo* = (CarName:**string** ,CarModel:**string**):**string** =>{  
 **return `Car Name :** ${CarName} **, Car Model :** ${CarModel}**`**;  
 }  
}  
  
**let *MyObj***= **new** ChildCar(**'BadawyCar'**,**'Model 2018'** ,220); *// Create An object from Car****console***.log(***MyObj***.*GetCarInfo*(**'MeboCar'**,**'Model 2020'**)); *// Call the Function From the Class by Obj of Car*

#######################################################################  
Access Modifier

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

Public : ده عام اى حد معدى فى الكود يقدر يستدعيه من جوه الكلاس من بره الكلاس

Protected : يتم استدعائه فقط داخل الكلاس الاب او ابنه الوارث منه

Private : يتم استخدامه فقط داخل الكلاس بتاعه و لكن فى طرق خاصه بتخلينا استخدمه بره و هى تمرير القيم ولكن حليا هو مش بيتم استخدامه الا داخل الكلاس بتعته بس

#######################################################################  
اختصار كود الكونستراكتور و البروبرتى الى بيستخدم فى تمرير القيم

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

**class** Car {  
 **public Name** :**string** ;  
 **public Model**:**string** ;  
 **public Speed**:**number**;  
  
 **constructor**(Name:**string** ,Model:**string** ,Speed:**number** )  
 {  
 **this**.**Name** = Name;  
 **this**.**Model** = Model;  
 **this**.**Speed** = Speed;  
 ***console***.log(**`Name :** ${**this**.**Name**} **, Model :** ${**this**.**Model**} **, Speed :** ${**this**.**Speed**}**`**);  
 }  
}  
  
**let *MyObj***= **new** Car(**'BadawyCar'**,**'Model 2018'** ,220); *// Create An object from Car  
// output => Name : BadawyCar , Model : Model 2018 , Speed : 220*

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

الكود المختصر و يعطى نفس النتيحه انى بشيل البروبرتى و احطها دايركت فى الكونستراكتور و على فكره هتقدر تستخدم البروبرتيس دى عادى جدا بعد كده كانك عاملهم عادى خالص (يفضل استخدام الكلام ده فى التايب سكريبت يا اسطى )

**class** Car {  
**constructor**(**public Name**:**string** ,**public Model**:**string** ,**public Speed**:**number** )  
 {  
 ***console***.log(**`Name :** ${**this**.**Name**} **, Model :** ${**this**.**Model**} **, Speed :** ${**this**.**Speed**}**`**);  
 }  
}  
  
**let *MyObj***= **new** Car(**'BadawyCar'**,**'Model 2018'** ,220); *// Create An object from Car  
// output => Name : BadawyCar , Model : Model 2018 , Speed : 220*

#######################################################################  
Modules (Export & Import )

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

بص ياعم الهدف من اكسبور و امبورت اننا نقسم و ننظم المشروع بشكل جميل و هى ذى فكره النيم سبيس بالظبط بحيث تكتب الكلاسات فى مكان ما و تعمله اكسبورت و بعد كده تيجى فى ملف تانى خالص تعمله امبورت و تستخدمه

و ليكن المثال الاتى   
1- اعمل ملف باسم الكلاس و اعمله اكسبورت

2- اعمل ملف باسم run و جواه اعمل امبورت للكلاس الى انت عامله اكسبورت ده و شغله فيه

Car.ts

ملف التايب سكريبت بتاع كلاس العربيه

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

**export class** Car{  
 **constructor**(**public Name**:**string** ,**public Model**:**string** ,**public Speed**:**number** )  
 {  
 ***console***.log(**`Name :** ${**this**.**Name**} **, Model :** ${**this**.**Model**} **, Speed :** ${**this**.**Speed**}**`**);  
 }  
}

MyTSFunc.ts

ملف التايب سكريبت هنادى فيه من كلاس العربيه

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

**import** {Car} **from "./Car"**;  
**let** MyObj= **new** Car(**'BadawyCar'**,**'Model 2018'** ,220);  *// output => Name : BadawyCar , Model : Model 2018 , Speed : 220*

هنا عملت اكسبورت و امبورت للكلاس ممكن على فكره اعمل امبورت لفانكشن عادى و مش لازم اجيب الكلاس كله و هكذا بقه ابقى شوف الى انت عاوزه بعدين بقه

#######################################################################  
Decurators with out Params

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

الديكوريتورس دى عباره عن فانكشن بتضيف خاصيه جديده للحاجه الى جايه قبلها

يعنى لو الديكوريتور قبل ال Class يبقى الخاصيه دى هتنضاف للكلاس و لو قبل function فالخاصيه دى هتنضاف للفانكشن و هكذا

هى عامله كده ذى ال attribute فى ال ASP.Net

تعالى نشرح بقه الديكوريتور على الكلاس بمثال

أول حاجه هتروح فى فايل ال TsConfig.json و تخلى ال expermentDecurator ب true

    "experimentalDecorators": true,

بعد كده فوق الكلاس هتكتب @اسم الديكوريتور و انا هنا هسميه Msg يعنى هيبقى فوق الكلاس مكتوب كده @Msg

و بما ان ده ديكوريتور فالمفروض هعمله فانكشن

و بما انه هيتشغل على الكلاس اذا انا كده المفروض اخليه يتعامل مع الكلاس فهبعتله فى البراميتارز بتعته constractor من نوع فانكشن و جواها بقه هضيف للكلاس بروبارتى اسمه Msg عن طريق الكوستراكتور و البروتوتايب فهيتكتب كده و خد بالك انا كل الى انا مكبر حجمه ده هو ده ال Decorator يا صاحبى

**function Msg(constructor:Function){**

**constructor.prototype.Msg = "I'm Decorator";**

**}**

**@Msg**

export class Greeting {

**[x: string]: any;**

    name: string;

    age: number;

    constructor(name: string, age: number) {

        this.name = name;

        this.age = age;

    }

     Hey() {

        console.log(`Hey ${this.name} - age is ${this.age}`);

**console.log(`Decorator is : ${this.Msg}`);**

    }

}

كده انا شرحت اول حاجه الديكوريتور الى مبيخدش براميتارز معاه تعالى بقى نشرح الى بياخد برامتيارز يا رجاله و هيبقى التغيير فى الكود بسيط المهم انك فهمت هو بيتعمل اذاى و ايه لازمته ذى ما قولتلك هو بيزود خصائص على الحاجات الى بيتحط فوقها مش اكتر و ذى ما انت شوفت فى المثال الى فات انا زودت على الكلاس بروبارتى جديده اسمها Msg و على فكره انا ممكن اسميها اى اسم تانى و ممكن ابعت جواه مليون حاجه مش مجرد بروبارتى واحده بس

تعالى بقى نشرح الى بياخد برامتيارز يا رجاله و هيبقى التغيير فى الكود بسيط و هو انى هخليه يقبل برامتيارز من نوع Generic عشان يقبل اى نوع من انواع البراميتار لما اجى ابعت فيه

#######################################################################  
Decurators with Params

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

ده المثال و هنشرحه

function classDecorator<T extends {new(...args:any[]):{}}>(constructor:T) {

    return class extends constructor {

        newProperty = "new property";

        hello = "Override hello from Decorator";

    }

}

@classDecorator

export class Greeting {

    [x: string]: any;

    property = "property";

    hello: string;

    constructor(m: string) {

        this.hello = m;

    }

    Hey(){

        console.log(`${this.hello} to  Old ${this.property} and to ${this.newProperty}`);

    }

}

بص يا سيدى ده ديكوريتور بيقبل البراميتار بس انا مبعتلوش براميتارز برضوا و عشان تفهمه بقه هقولك على شرح الفانكشن الى اسمها classDecorator دى

بص يا سيدى دى فانكشن بتقبل اى كميه من البرماتيارز و كذلك من اى نوع عارف ليه لانى قولتلها انا عاوز البراميتارز الى تجيلى تبقى من نوع ال T الى هو Generic يعنى اى حاجه و قولتله اورث بقه البراميتارز الى هيجولك دول و الى هيبقوا فى اوبجيكت و انا هفرط الحاجات الى جايه دى عن طريق ال … ال array spread و ده على اساس انى لو بعتلك لكشه براميتارز فهو يفهم انها ذى الاراى كده فيخدهم يفرطهم و يوزعهم

طيب الفانكشن من جوه بقه انا بدل ما اتعامل مع ال prototype قولت ليه مخليش الريتيرن يبقى عباره عن كلاس بيورث من الكونستراكتور بتاعت الكلاس الى ال Decorator محطوط عليها و جوه الكونستراكتور ده العب فيه براحتى يعنى عاوز ازود بروبرتى او اعمل override على اى حاجه اعمل عادى كله فى ايدى خلاص و بعد كده اقدر استخدم الديكوريتور عادى خالص بقه يا صديقى

طيب بعد التجربه دى انا عاوز اوريك بقه لما ابعت براميتارز او ارجيومنتس فى الديكوريتور بقه فشوف المثال ده

#######################################################################  
Decurators with Params

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

انا لقيت ان المثال الى فوق ده صعب فى انى ابعت برامتيار بشكل سهل و كده فقولت نجيب اسهل حاجه ذى ما الناس بتعمل

function classDecorator(Config: any) {

    return function (target: any) {

        Object.defineProperty(target.prototype,'Address' , {

            value: Config.Address

        } );

        Object.defineProperty(target.prototype,'Name' , {

            value: Config.Name

        } );

    }

}

@classDecorator({

    Address: "Cairo",

    Name:"Mahmoud Badawy"

})

export class Greeting {

    [x: string]: any;

    constructor() {

    }

    Hey() {

        console.log(`Hello ${this.Name} from ${this.Address}`);

    }

}

ذى ما انت شايف انا عملت الديكوريتور بيقبل بارميتار اهه و رحت استقبلته فى كونفيج و بعد كده جوه فانكشن الديكوريتور عملت ريفاكتور فانكشن و جواها كل مهتها انها تضيف بروبارتى للاوبجيكت عن طريق التارجيت الى هو أوبجكت الكلاس الى انت حاطط الديكوريتور فوقه من الاخر و الى بضيف عن طريقه البرووبارتى عن طريق البورتوتايب عن طريق فانكشن ال defineProperty الى بتاخد 3 حاجات اولهم البروتوتايب الى هتحط عليهو بيبقى بروتوتايب التارجيت الى هو اصلا الكونستراكتور او اوبجيكت الكلاس الى انت حاطط الديكورتورفوقه و تانى حاجه اسم البروبارتى بتاعك و بعد كده 3 حاجه القيمه بتعت البروبارتى ده و ده بيبقى داخل أوبجيكت

كده شكرا ال Decorators تم شرح حاجات بسيطه فيها يعنى يعتبر اتشرح فيها اهم حاجات و خلاص خش اتعمق انت فيها من هنا بقه

<https://www.typescriptlang.org/docs/handbook/decorators.html>

نسيت اقول ان السطر ده بتاع الانديكسر مهم جدا عشان الديكوريتور يشتغل صح

   [x: string]: any;

و ده شرح حلو للديكوريتور برضوا

<https://itnext.io/a-minimal-guide-to-ecmascript-decorators-55b70338215e>

أو ده

<https://gist.github.com/remojansen/16c661a7afd68e22ac6e>