**Cons that drive Javascript  to typeScript:**

Pitfalls (typescript = vanilla javaScript)

**Costly freedom :**

Don’t know proper idea how to call a function when number of files of projects grows. Since in a function we don’t know parameter types and description datatype since it is a issue.

**Commenting :**

No facility to formalize description about code purpose. Such as commenting we give to know logic

Maintain hundreds of file through commenting is really challenging to maintain large codebase and its logics.

**Weaker developer tools :**

Not have builtin ways to identify types.

Difficult to autmomate large changes.to gain insights about code changes.

***Loose Documentation:***

No description to formalize about code purpose.

Js doc file used by developers key issues often make it unpleasant to use in large codebases.

**TypeScript :**

In 2010 typscript made by Microsoft. Specifically featured type sensitive language.

It is the superset of javaScript including EcmaScript (ES6).

When writing code if there is any issue it prompts error called type checker.

Contains existing based js syntax alongwith addition to typescript specific syntax defining.

**Compiler** : type checker runner reports any issues.

**Transpile** = compiler based machine readable code.

**Language services** : a program that uses type checker to tell code writers such as vs code to provide helpful utilities to coder.

It cant recognize between method or function at code written time.but shows error at runtime

Text

Description automatically generated

We can runs thing as above.

***Code sensitiveness***

***01***

Like :

Const fname = “message”;

Const namelength = fname.length(); // error in js’

***Freedom through restriction:***

Change from taking in two parameter , required one parameter , but give at call time two parameters which is given error while code written in typescript but no error will be shown in javascript.

Graphical user interface, text

Description automatically generated

Example 02 :

const fname = "message"; // legall illegal variable error

const namelength = fname.length; // error in js

console.log(namelength);

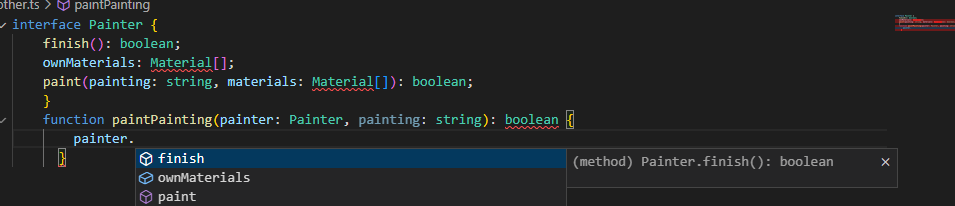
***02***

***Precise documentation :***

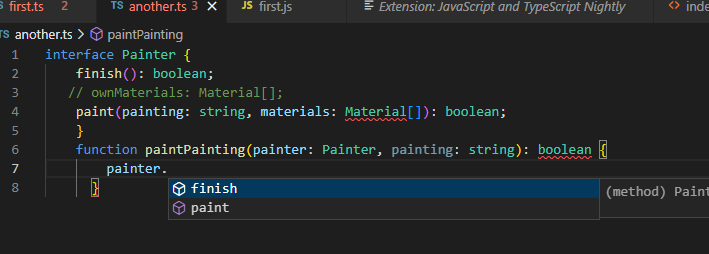
A TypeScript developer reading this code for the first time could understand that painter has at least three properties, two of which are methods. By baking in syntax to describe the “shapes” of objects, TypeScript provides an excellent, enforced system for describing how objects look

It can define objects at runtime if we made changes like commenting some of the properties we fetched out from other functions.

shows things at code :



Now if we made change it will immediately show upon calling.



Stronger developer tooling as shown above.

***03***

Type script provides ***autocompeletion*** suggestions at many ends such as there in string.

Graphical user interface, text, application

Description automatically generated

**Development Environment Setting:**

To install run  on terminal :

npm i -g typescript

check typescript :

tsc -v  (cmd)

install packages for a project to run ts :

npx tsc --init

to compile ts file :

npx tsc first.ts

we cant run ts file on webpage using html , needs to convert it to js file to run ts file.

***04***

**Parameter naming:**

Showing issue here : difining it as a number must.

function addTwoNumbers(value1,value2){  // define it showing as compliant

    return value1 + value2   //returns paramerters

}

console.log(addTwoNumbers(2,2)); // passing arguments and callout function

here see another issue:

function addTwoNumbers(value1 : number ,value2: number){  // now we defined a number no problem here

    return value1 + value2   //returns paramerters

}

console.log(addTwoNumbers(2,"abc")); // passing arguments and callout function , we gave string instead of number

//shows error at code written time.

Showing error with red mark.

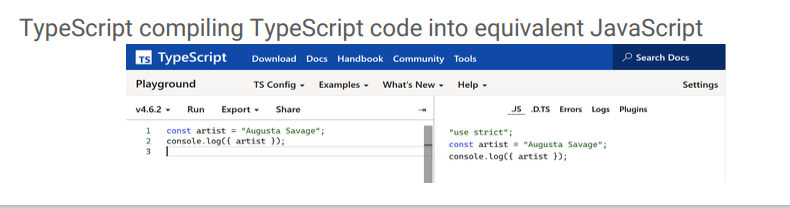
**Flows of typescript:**

* Not try to give solutions while suggestions.
* Only compile on ecma script proposals
* Preserve runtime behavior of javascript code.
* Slower than javascript compile transfile to js file. Generally missaccurate or misleading.
* Browsers only nodejs wil run it.
* Remedy to bad code don’t give opinion overtime
* Takes all things pros and cons of js to ts.
* Finished evolving what typescript cant do , versions have bugs to overcome new versions introduced to overcome bugs.

***06***

**Compiling syntax:**

Using typescript playground allows you a feature to write typescript syntax simaultaneously developing js code on the right side of the screen.



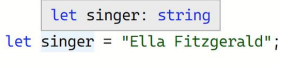
***What’s in type :***

It means a type value to be understands by typescript.

There are seven basic primitives are In typescript :

* Null (null named)
* ……..(undefined) letting blank space no code wrote if ever.
* True (Boolean)
* 1234(integer)
* 1.24(float)
* 1234…..N(bigint)
* Symbols

If you mouse over a variable it will names its datatype.



Ts knows ternary expression always results in a string. So we cant gave it type other than string.

Text

Description automatically generated

**Type Systems:**

It is a set of rules for how languages understand what type constructs may have.

Graphical user interface, text, application

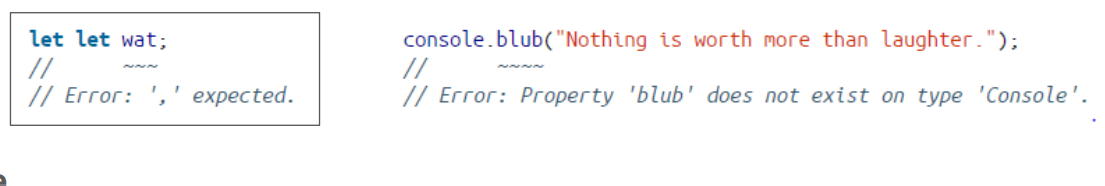
Description automatically generated

**Kinds of errors :**

Writing typescript you encounter two types of errors:

*Syntax :*

Placing gaps while file making of ts.

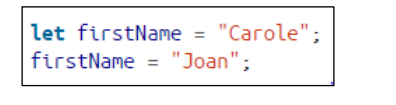


*Type:* type errors detects problem in your ts file’s types. Apart from syntax remain valid.

**Assignability:**

If you later on after compilation made change variable’s value like string variable value is change to another string. Its fine

But change datatype will cause an error throw.

 Text

Description automatically generated

Graphical user interface, text

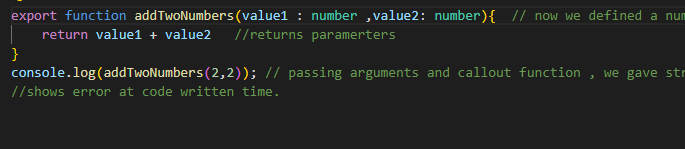
Description automatically generated

***Type Annotation:***

We can use properties , functions methods , variables of one ts file to another ts file through module method.

Script(main workspace ts file) and module(export file to main ts file).

Applying export to a function :



Importing the file’s function :

Text

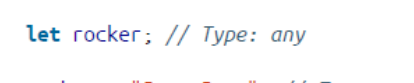
Description automatically generated

Here’s output :

Text

Description automatically generated

By default if we don’t initialize value to a variable typescript consider its type any.

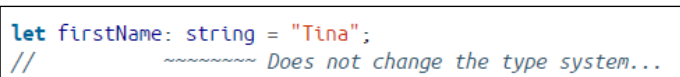


We can declare type to a variable without initializing value init like below :

Text

Description automatically generated with medium confidence

And afterwards we can assign value to a variable.

We cant use column ( : ) to a string defining on the same line while declararing it gives error. 

Specific type property is just availbe for an object if we try to access properties of that object.

Text

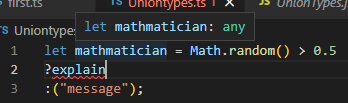
Description automatically generated with medium confidence

***Unions and literals :***

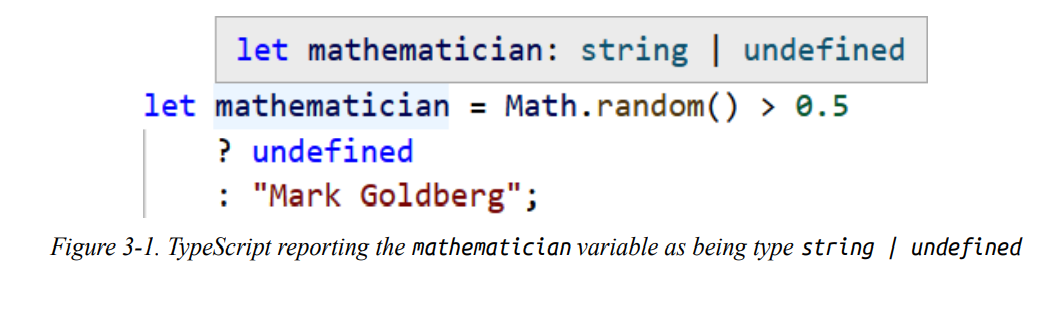
**Union types :**

**Math.random()** generates numbers between 0 to 1.

Type error detect defined check varibale is number while if checks for string.

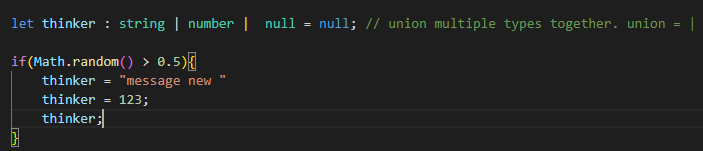


Look at another example :



it uses Pipe ( | ) operator to compare values types assigned with variable.

Type check error : Allowing three property simultaneously together.



If we violate union above defined , by violating sequence or datatype defined.

Check the error then

Graphical user interface, application

Description automatically generated

***Possible types Union Properties :***

Typescript will not allow use of member properties that exist on all possible types of union.

It will give you an error if you try to use properties that does not belong to your used variable type.

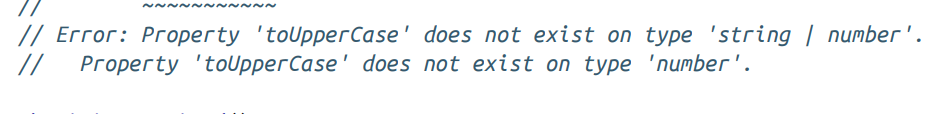
***Text

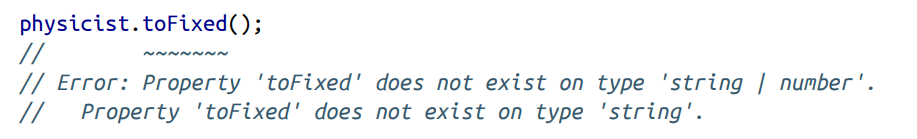
Description automatically generated***

Graphical user interface, text

Description automatically generated

We can convet number to string but cant used upper case it is the rule !.





***Narrowing :***

Ts knows value type specifically and display string relating methods and functions as suggestion particularly.

 similarly for number the same rule proceed.

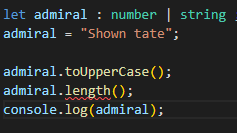
A logical check that is used to narrow types called **type guard**.

Ts narrow variable’s type to that value’s type.

Shows just methods functions and things that specifically related to the variable’s types.

**Assignment Narrowing:**

If you accidentally used methods or functions apart from number type for instance error will throw we are just restricted to use number functions methods with numbers just see below example :



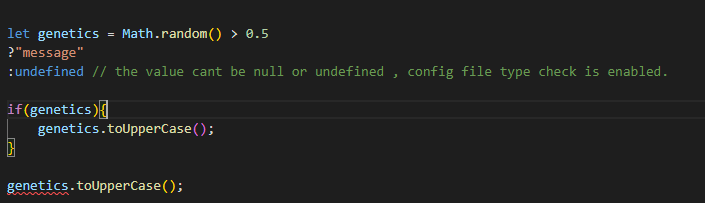
Text

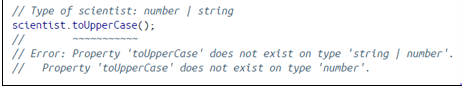
Description automatically generated

**Conditional Checks:**

Cheking on if conditions what is happening :

Type script is smart enough to understand the statement inside paranthesis is of what type.

****

****

Type of check :

Used typeof operator to narrow down variables type

Example 01 :

Graphical user interface, text, application, email

Description automatically generated

Example 02 :

Text

Description automatically generated

Checking if type is number given in variable’s as a value than if statement works fine.

Otherwise if we initialize value other than number if statement won’t work an error throw.

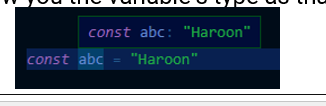
***Literals :***

Using let or var we tell compiler that overtime values stored inside any one of them might be modified.

On the other hand if we use const we tell compiler that its value remain same overtime.

Example 01 :

variable type shown as a literal when we declare a constant variable and hover mouse over it. A literal value type specifies specific type set of values.



Fixed values means it values cant be change later on for that we use const with variable declaration.

Example 02 :

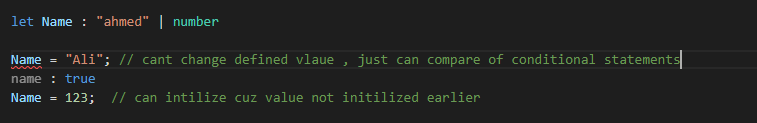
Text

Description automatically generated

In above example : a union of every possible matching literal value.

Where it doesn’t find defined literal an error throw.

Example 03 :



Union literals and other datatypes.

Example 04 :

Different literal datatypes within the same primitive data type are not assignable to each other.

Literals assignability error :

Example 01 :

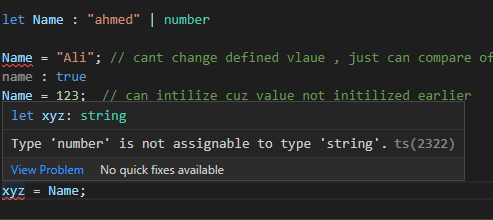
Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Example 02:



Example 03 :

Text

Description automatically generated

***Strict Null Check :***

A billion dollar mistake is allowing null values to be used in places where required a different type.

Cant give null or undefined value to aonther variables used. Means values cant be undefined or null at all cost

Can be edited null strict at config ts file.

Text

Description automatically generated

If i disable config gile type check



Error should be removed.

Graphical user interface, text, application

Description automatically generated

.

**Truthiness :**

Check variable if truthy before using it.

All values in js are truthy.

Except those falsy : 0 , -1 , null , undefined.

**Nulllify** :

Text

Description automatically generated

Text

Description automatically generated

Typescript can sense that variable under if braces is type *string.*

**Variable without initial value :**

Start using variable without assigning value in it throw error called strict null check.

Graphical user interface, text

Description automatically generated

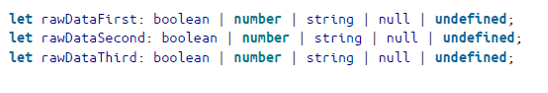
A screenshot of a computer

Description automatically generated

Ts can sense that variable before used with methods is undefined so there is no value over actions are desired to perform.

**Type aliases :**

A type aliase starts with the type keyword. Then we give name to the defined type and then conditions of data types allowed to used with variable declaration and usage.



Example 01 :

Text

Description automatically generated

Example 02

custom type used to defined aliase

Text

Description automatically generated

Since lilly isn’t defined.

Graphical user interface, text, application

Description automatically generated

Example 03 :

 combining type aliases

type id = number | string

type id2 = id | boolean | null  // one aliase terms used in another values by assign it to 2nd aliase.

Type aliases are like type annotations are not supported by javascript to be transfile or compile to javascript output.

They purely exist on typescript file system.

***Objects:***

Array objects , call  ,modify , replica.basically it is a type that is defined with attribtutes and information about the object.

And objects reuse afterwards.

Object contains properties and methods.

Property :

Graphical user interface

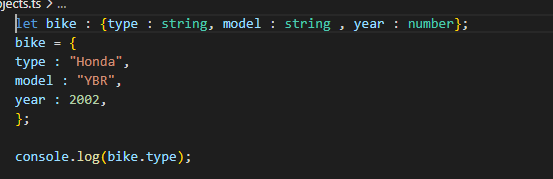
Description automatically generated with medium confidence

Methods :

A picture containing table

Description automatically generated

Method.



Example 02 :

Object as aliase.

Text

Description automatically generated

We can avoid repetitive type of object duplication by storing type checks into another variable. See above !.

See below now :

Graphical user interface, text, application

Description automatically generated

**Ts is structural language.:**

all of the values are considered to be equivalent type. If all of their component features are of same type.

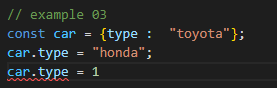
Telling ts compiler whatever objects you use are necessary to be of same type of property.

Example 03 :

Graphical user interface

Description automatically generated

Example 04 :



Graphical user interface, text, application

Description automatically generated

**Nominal type :**

Explicitly needs to call the type through object. (shows available properties of a type). In normal programming languages.

**Duck type :**

Programming languages can  be classified as duck type , explicitly needs to define type of an object , languages cant detect it by itself. Called duck type.

**Structural type :**

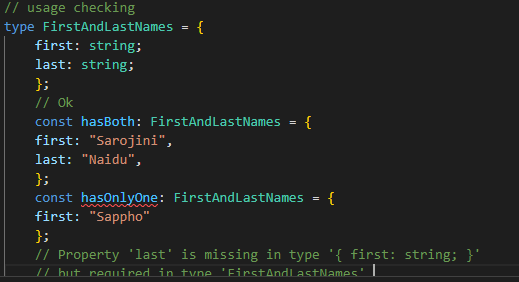
Typescript is structural typed language. Not require explicit declaration of the dependencies.

**Usage checking :**

The value assignable to the object type must be of defined type such as for string we just gave assigning value in the form of string.

If any require of the object type is missing in the object typescript will issue an error.

If we define const then we cant change variable type and cannot its value can be change later on.



We cant change value of constant defined variable.

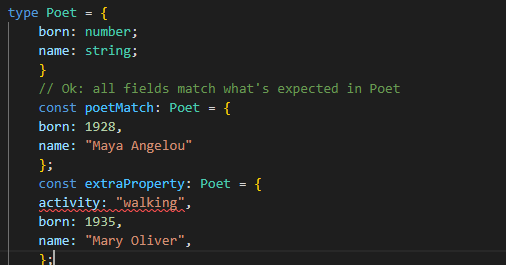
Graphical user interface, text

Description automatically generated

***Excess property checking :***

If a variable is declared with an object type and its initial value has more fields than its type described.

We cant use extra defined property if its is not initially defined in the object property making.

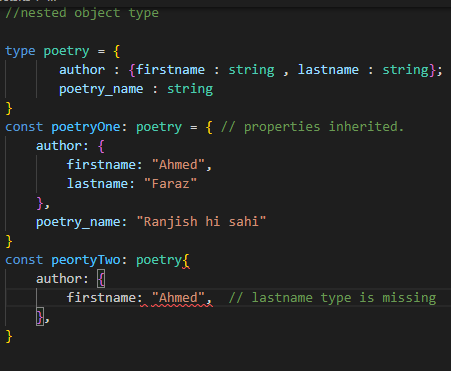


Text

Description automatically generated

Shows activity isn’t avaible in the type poet.

**Nested object type :**



If inheriting properties from one type , we need to use all the properties mentioned in the type then must.

**Optional properties:**

Objects types somewhere require must and somewhere we make the property use optional which wont disturb an error message if we wont use that particular property.

Text

Description automatically generated

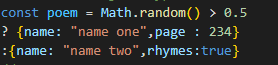
**Unions object type :**

Automatically makes its type union.  A type that can be described in one or more different object types that have slightly different properties.

**Infrared object union types :**

If a variable has given an initial value could be one or multiple object types.

We can add ( ? ) and ( : ) in a type’s property to run things as optional properties as we want.

And we can optionally call any type that is required by us.  Text

Description automatically generated Graphical user interface, text

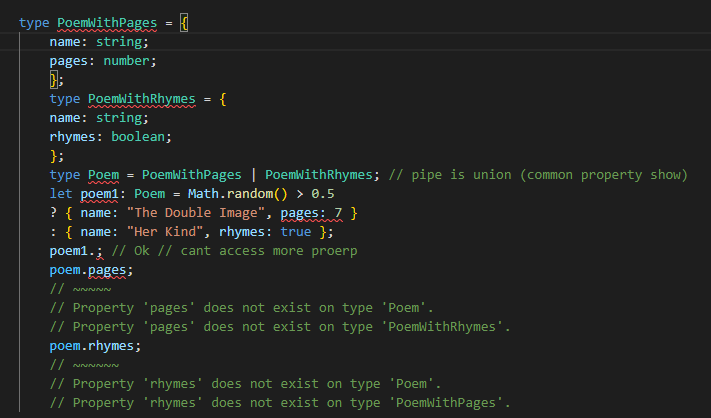
Description automatically generated

Poem always have the name properties in the object type and may or may not have type object rhytem or pages

**Explicit Union Types:**

Union pick common properties of two objects. And just show common properties and unique properties between two obecjts can’t be access.

Merging two objects together. And common type object is shown.



Graphical user interface, text, application

Description automatically generated

Only shows common property

**Narrow object types :**

Text

Description automatically generated

It will narrow the union type to only constituent to the contain property.

Peom with pages narrowed in if

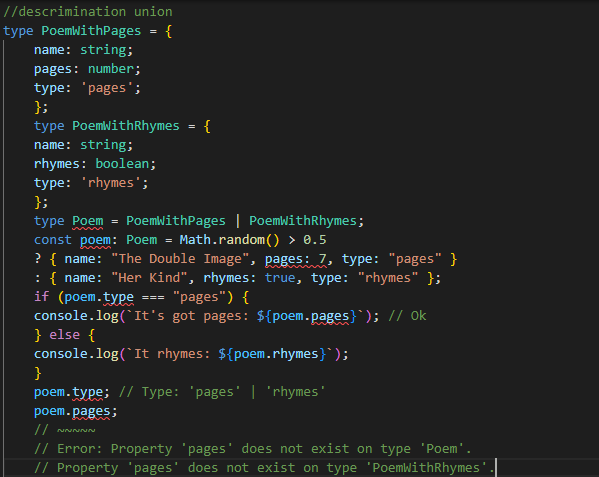
Poem with rhythems narrowed in else.

**Descrimination union:**

Literal types that can be used to narrow down possible current type.

This is called descrimination union.

The property whose value is indicates object type called descriminant.

 Text

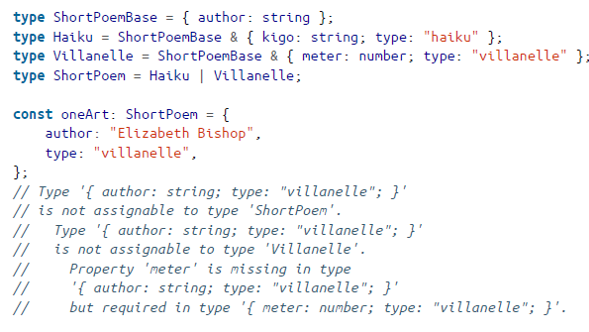
Description automatically generated

**Intersection types :**

Ts allows to combine multiple types together of object and use it in some variables.

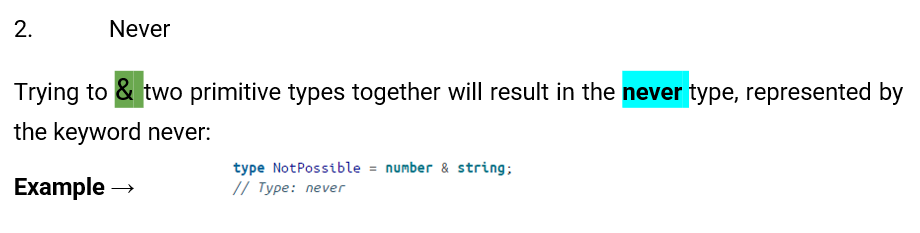
Combing that type’s properties we can through  (&) operator.

**Long assignability error :**

****

Mixing up use can make things messy which is either common which is used all object type property.

**Never problem:**

****

***Nested Objects type:***

move nested obects into their own type name to make the code more readable for more readable type error message.

Text

Description automatically generated

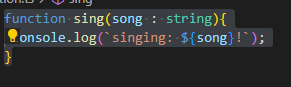
Since the poem match initiliazation into the values of properties, into a sequence as the  properties defined above.

While in poemmismatch we cant follow sequence to initialize properties.

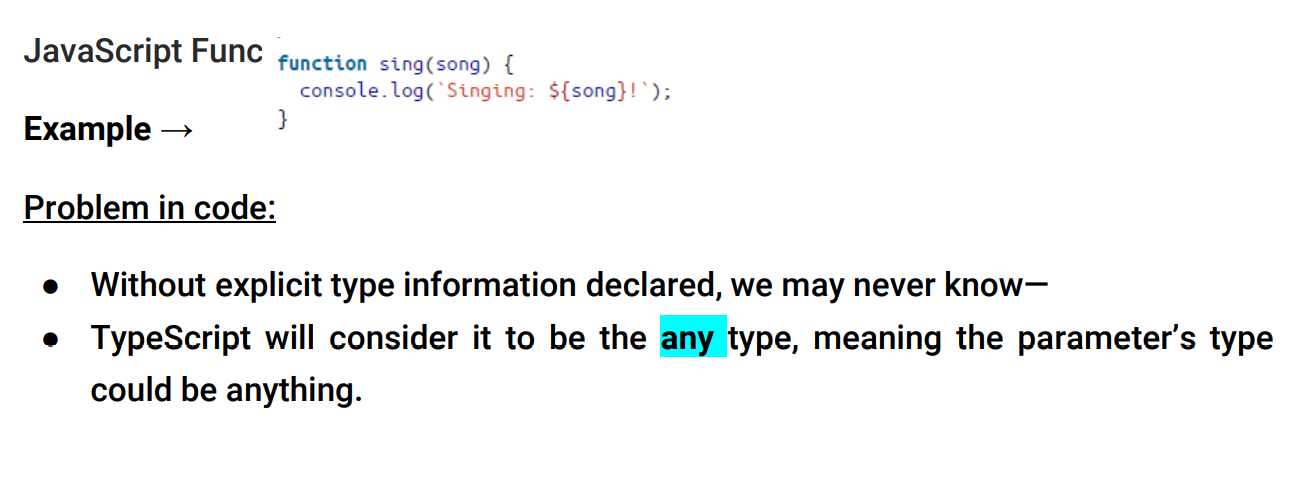
***Function:***

Type needs to defined.

Here we use string.



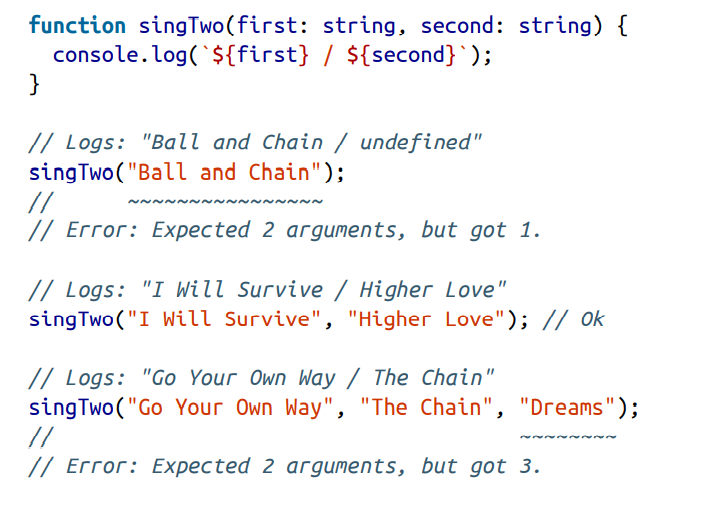
Example 02 :



**required parameter:**

We needs to use one argument if givent we can just give one parameter at a time

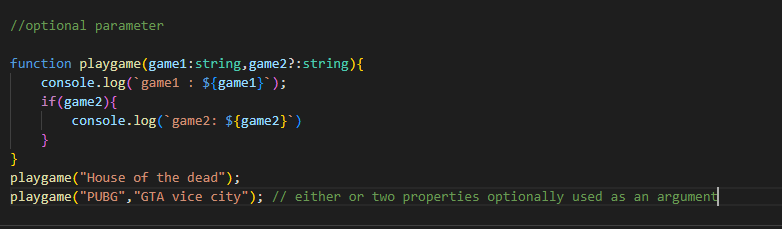
If given too few or too many arguments nonetheless how many were defined.

 Graphical user interface, text

Description automatically generated

***Optional parameter:***

Optionally defined parameter either use or not as below you can see:



***Function default parameter:***

Ts offers default value initialized before we proceed to code function. It given value through use of ( = ) and a value in their declaration.

***Text

Description automatically generated***

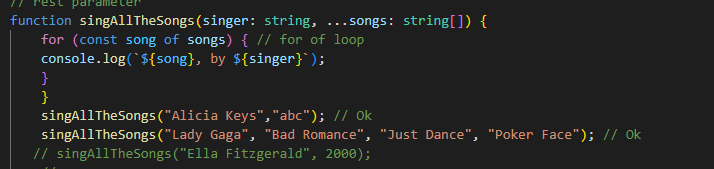
A screenshot of a computer

Description automatically generated with medium confidence

***Rest parameter :***

We can save multiple values in one variable parameter.

Sometimes functions may be called number of arguments this way we can save multiple values into an array and print them using any loop method.

 Text

Description automatically generated

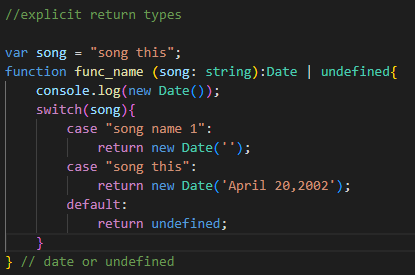
note : it is must to use same dataype in the array otherwise error will be thrown.

***Return types :***

Text

Description automatically generated

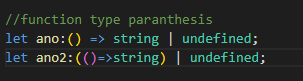
**Explicit return types:**



**Function type paranthesis:**

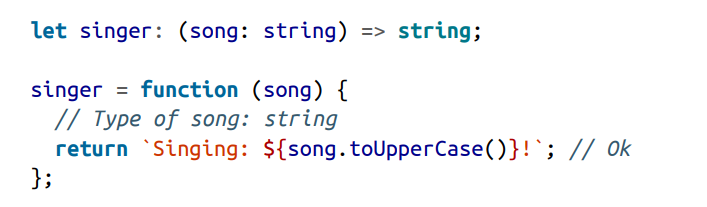
Function can be given as argument or as pipe union method

Function types Can be placed anywhere that would be used.

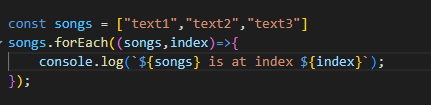


**Ts infer the types:**

TypeScript can infer the types of parameters in a function provided to a location with a declared type.

****

**Parameter type inferences:**

 Text

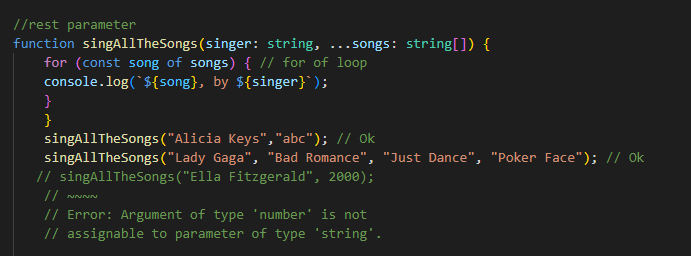
Description automatically generated

Function types passed as arguments of parameters types have their own parameter types inferrad as well.

We cam loop through number of values of array values through function and similalry without function as well.

For function agrumeent type should be same as array type.

Example 02:

Text

Description automatically generated

If we add datatype apart from string then error will be thrown.

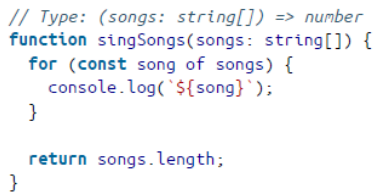
A screenshot of a computer

Description automatically generated with medium confidence

**Return Types:**

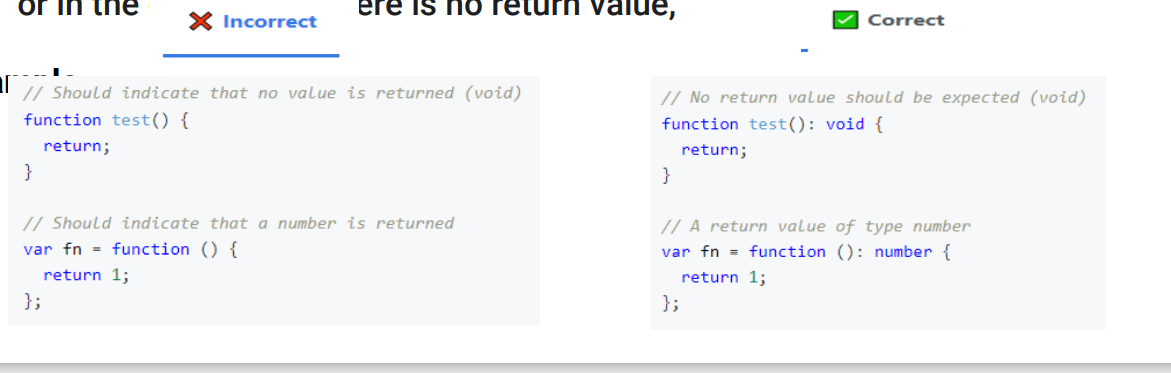
Ts can understand which return type is possibly gonna return.

If function returns more than one values then ts infer all possible return values and union them and eventually give combine return result.

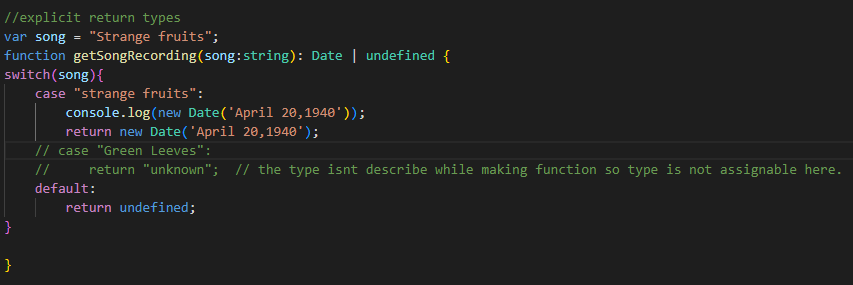


Explicit return type :

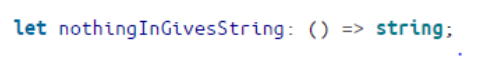
Give a return value’s datatype  this ensures exact possible return value will be given in output.



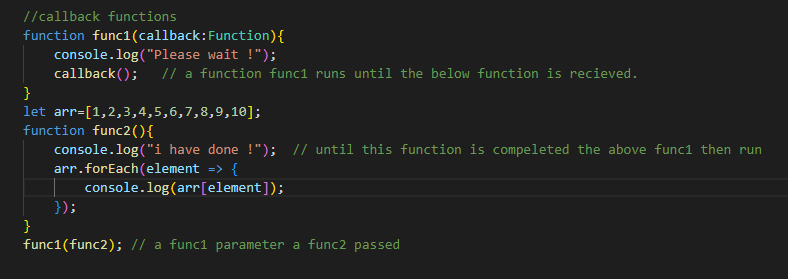
Example 02 :



An arrow function is not a type instead a body of a function which can be used by other function anywhere overtime..



***Function callback:***



We can use callback whenever our backend function working into the frontend the other function work alongwith until the backend function complete.

A function passed into another function which is invoked by the outer function.

***Function type parenthesis:***

Variable type union we give more than one.

whenever we require from a function (arrow function) to be used for more than one type of data.

Graphical user interface, text

Description automatically generated

***Function type inferences:***

***ts can sense the type of data we input. here index parameter is inferred by ts automatically.***

 detect it is a string type

Graphical user interface, text, application

Description automatically generated Text

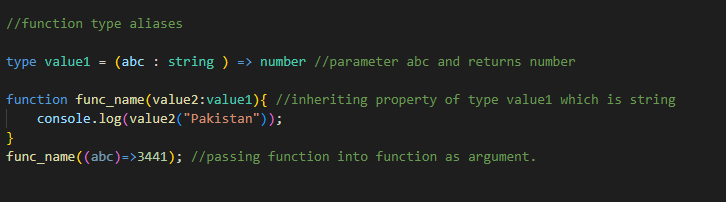
Description automatically generated

***Function type aliases:***

Type makes alias of a function.

We can use type’s function type anywhere we wnna resuse.

alias = adding more than one type to be used by the variable or functions.



Text

Description automatically generated

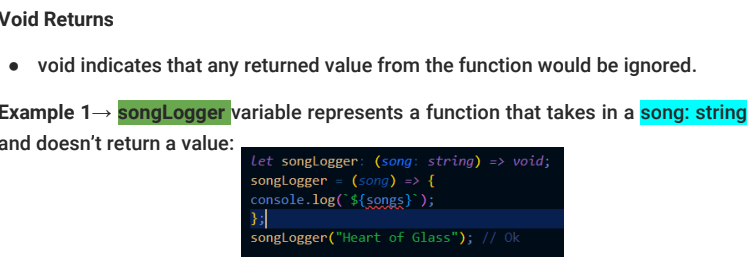
***Function type void:***

Functions are meant to return nothing value by default.

It won’t return nothing if return type is not used

Return type is mentioned but not used it is consider void then. Type remain void.

by using type with a void declaration that means if this type is used by any function any value from it would be ignore to be returned or used for other function.

example 01:

example2 : (void returns).

Graphical user interface, text, application

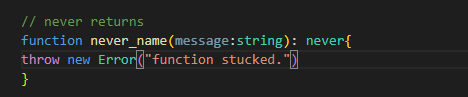
Description automatically generated

**Never returns:**

Returns a message whenever a function called doesn’t work.

a never type contains no value.

it is used by other function whenever an error found out in other functions , this function is than invoked and displays whatever message is written init.



**Function overloads:**

Wnna use same variable in multiple function wont write new variables again and again. But to use overtime into another functions.

A function name overloaded with different jobs. At multiple functions.

Making same name function many times as well.

Text

Description automatically generated

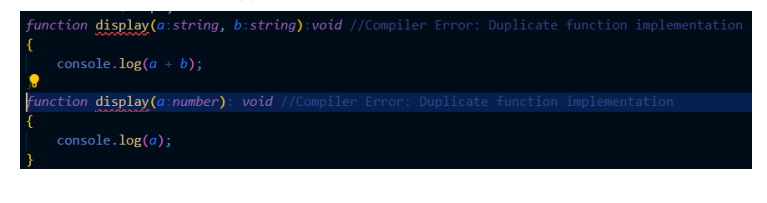
Text

Description automatically generated

Cant defined with same name duplicate if it is not a overload defined.

Example 02:

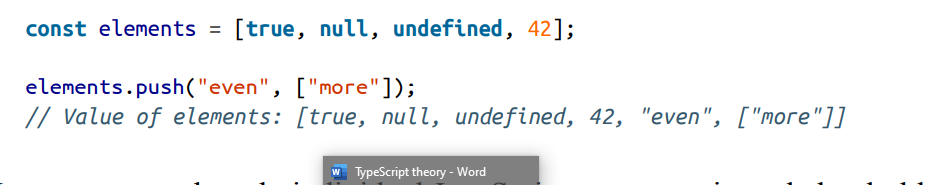
a function overloading with different numbers of parameter is considered duplicate.



***Array:***

***arrays are widely flexible and hold mixture of elements as their values.***

**Simple array:**



***array and functions types:***

array can be used as a type that can later on used by any function:

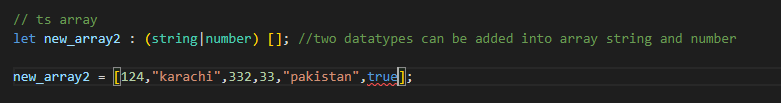


**Ts array union type array:**

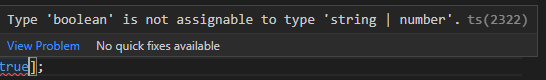
we can use multiple types to indicate that array elements can be of more than one type of elements init.

that below screenshot means that type is either be string or number.

example 01:

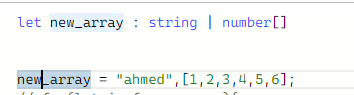
******

Two types of array are only allowed string and number just other types wont allowed.



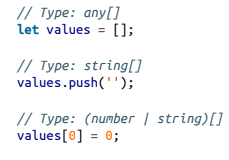
example 02 :

type is either an string or array of number.



**Evolving any array :**

**if a variable that is initially defined dont include type annotation(string , numbers etc) than it will treat as any type by ts.**

****

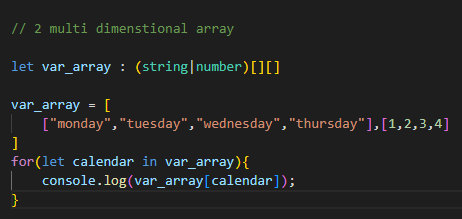
**Ts array type string , array of numbers then array of boleans:**

Text

Description automatically generated

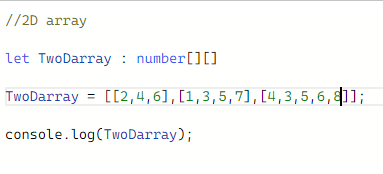
**2 dimensitonal array:**

2D array or multi dimensional arrays have more than one “[]” in their type defined.

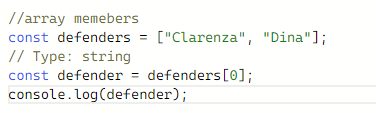
 Text

Description automatically generated

example 02 :



**Array members:**

accessing members elements of an array through index number 

**Unsound member:**

Text

Description automatically generated

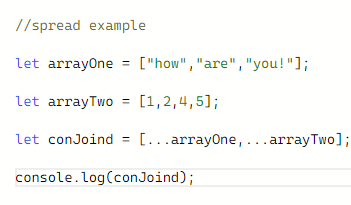
sometimes typescript is not guess the type defined right. we as a reader have to be careful to identify that type of unsound errors in compile time.

**Spread:**

array can be used to joined with another array of other variable together through

….(this indicates spread variable) , whereas this two arrays are joined to be represented as a single unit of array at runtime.

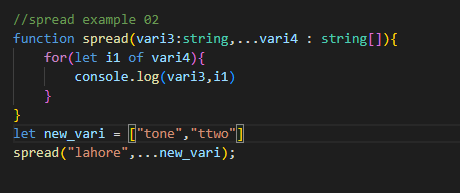
Example :



**Spread rest parameter:**

array can be used as an argument to be called as rest parameters.\

where we used spread method to specifically defined that type to be used properly.

 Text

Description automatically generated

**Tuple:**

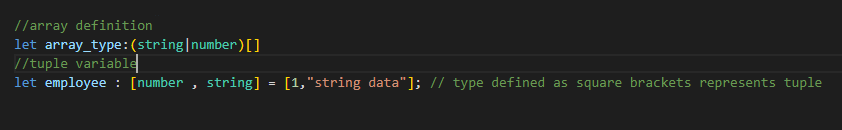
Ts introduced a new data type called tuple.

We have to give data and type defined in square brackets with (=) between type defined and initialization.

 rather than defining type in simple brackets then define initilization.

tuple arrays are of fixed size and have specific known index for each index of possible elements in array.

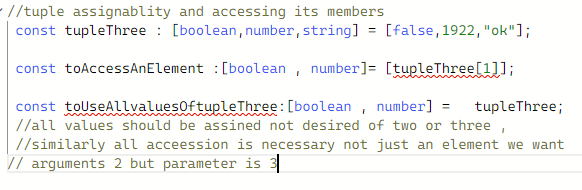
**Tuple and array difference:**



List of fixed elements where a number of length of elements is limited.

**variable assignability:**

sequence of initialization matters again , number of parameters should be equal to number of passed arguments when calling a function for initializing values init.

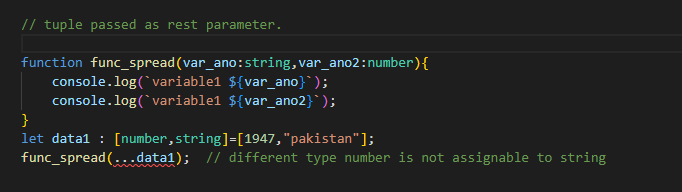


**Tuple passed as rest parameter:**

since tuple type is more strict with variable length of an array types.

it is more needed to be alert about sequence of initializing values to the types and variables defined.

tuples of different length are not assignable to each other.



since in above example initially we see type string is first then number.

hereafter type is number than string,

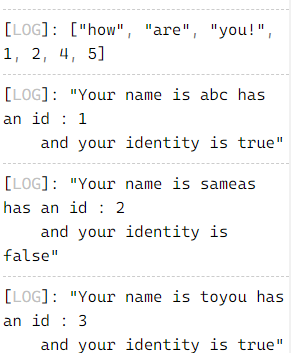
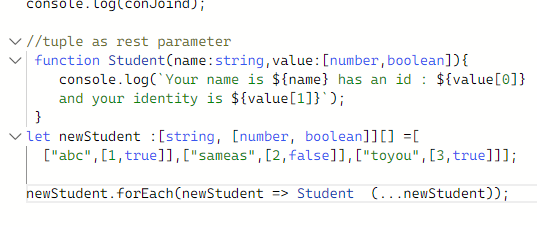
upon the accession of data1 spread is must be defined to be string than number is necessary.

to correct above error :



sequence matter alot.

example 02 :



**Type Inferences:**

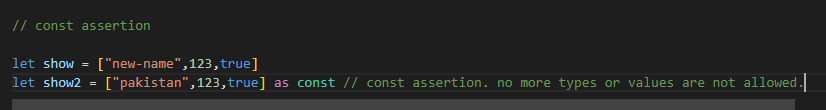
ts treat created arrays as variable length array and their values are of flexible length. rather than a fixed length value tuple.

to be more specific apart from array to make it tuple we have two methods const assertion and explicit tuple types.

**Const Assertion:**

turning it flexibility of adding more values or changes to the defined show2 variable than turning it to be behaved like tuple and restrict to fixed length values.

permit changes later on.



***Interfaces :***

We resuse methods and object’s properties into many classes ,

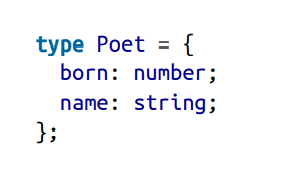
Interface works as an contract and contain no initial values but its predefined policies and structural standard which can be follow on and use by other classes.

Object whereas user defined datatypes.

Help to customize user defined datatypes.

Similar to type aliase.

We know type alias as :



Similarly if we convert it into interface its look like as:

Graphical user interface, text, application

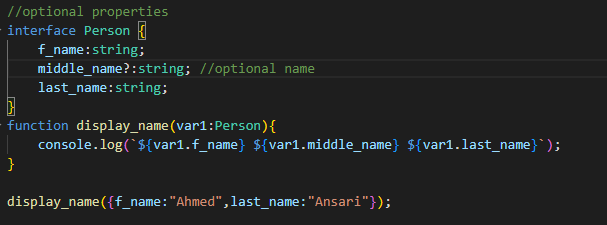
Description automatically generated

However, there are a few key differences between interfaces and type:

* Two interface can be merged together.
* Interface can check type of structure of class declarations while type alises cant check data types used in the class structure.
* Interfaces considered to be in object named. Rather than alias which are un named and declared with type keyword.
* Interfaces are generally speedier for type script checker.

**Optional properties:**

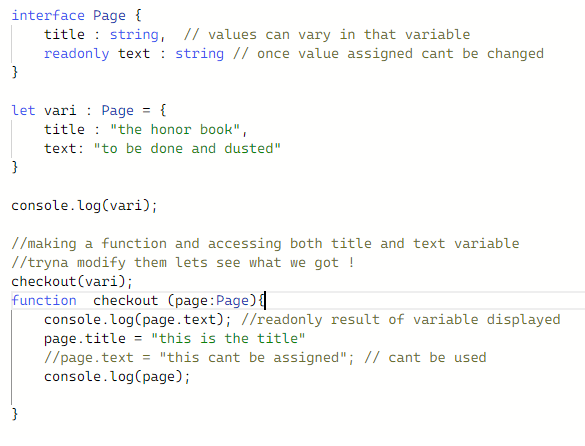
**it is the same as we did earlier in array methods we declare some of the variable optional which can either or not used as per needs and wants of the user.**

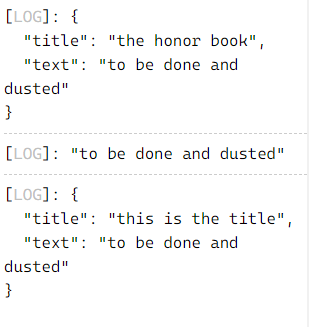


**Read Only example:**

**after declaring a variable readonly once defined value , cant be change altered but can be just access for results stored on readonly declared variable.**

**example :**

****

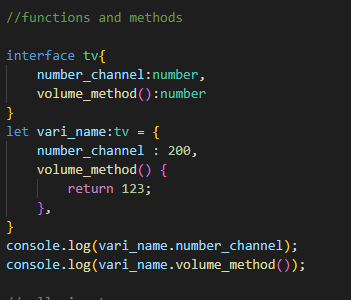
****

**Functions and methods:**

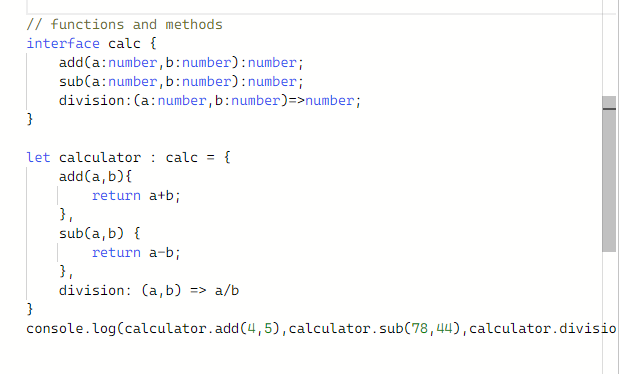
**functions and methods can be called into the interface with parameters into (functions methods) , later on we develop methods or arrow functions and apply logic in their body used them accordingly.**

**method syntax: a function can be defined and intended to be called as a member object.**

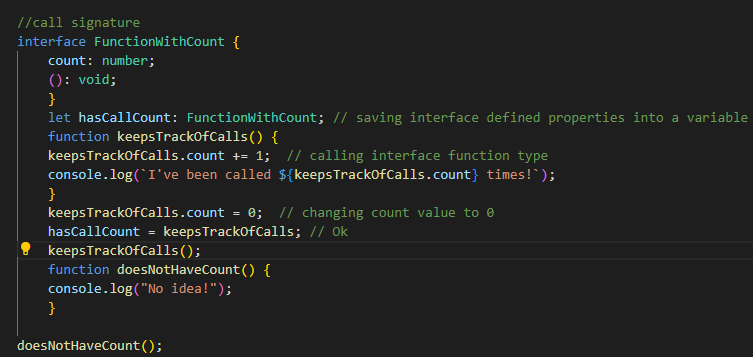
**function syntax: declaring a function can be intended to be called as standalone function for using its value and logic defined init.**



example 02:



**Call signature:**



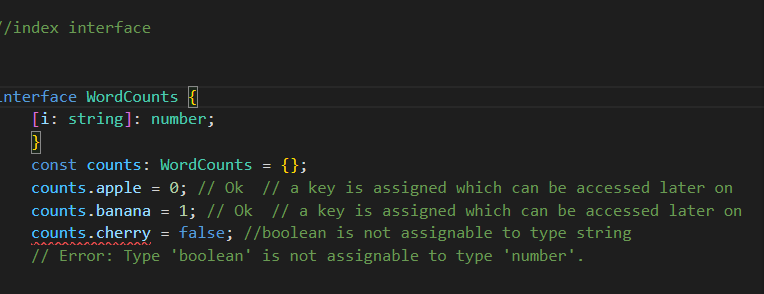
**Interface index:**

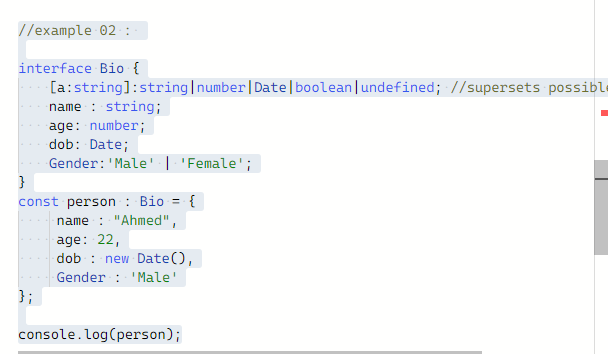
**we can use a specific syntax alognwith interface , called interface index that can specifically assigned a key with an array elements used for identification of that elements uniquely and dynamically assigned desired key to desired element.**

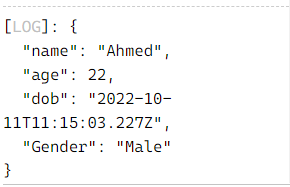
**here they are most commonly used with string keys.**

**object should bring back a value of that string key no matter which property is accessed**

**assigning values to the objects but they are not conveniently type safe.**

****

****

****

**numeric index signature:**

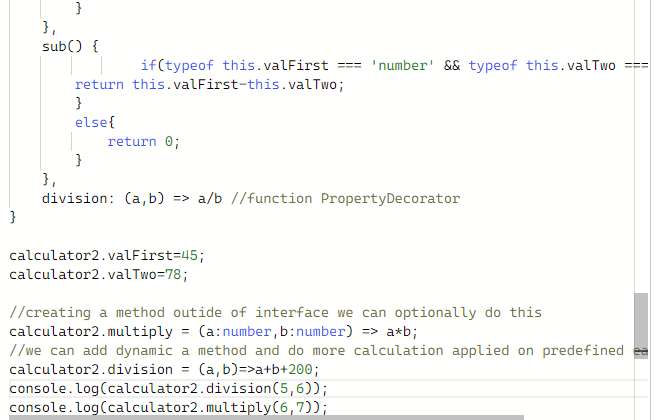
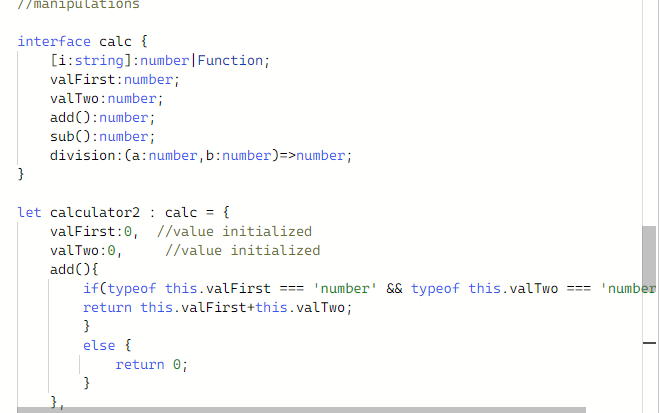
**we does this work to narrow down our method/function to be used which is used to restrict the input user to only allow numbers as keys for an objectTypeScript index signatures can use a number type instead of**

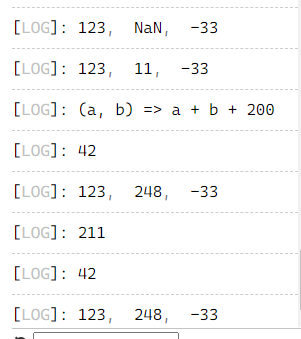
**string but with the same catch as named properties that their types must be**

**assignable to the catchall string index signature’s.**

**to be fetch as input for a defined function/method.**

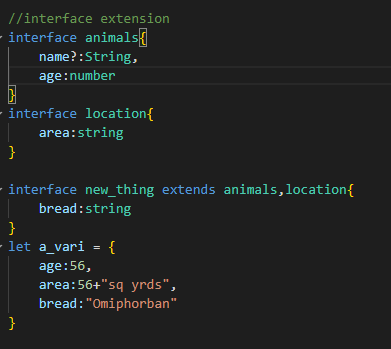
**example :**

****

-****

***Interface extension:***

We can use two or more separate interface and fetch their features into other functions and into other interfaces. This is used when we want to resuse codes written in other interfaces.



***Interface merging:***

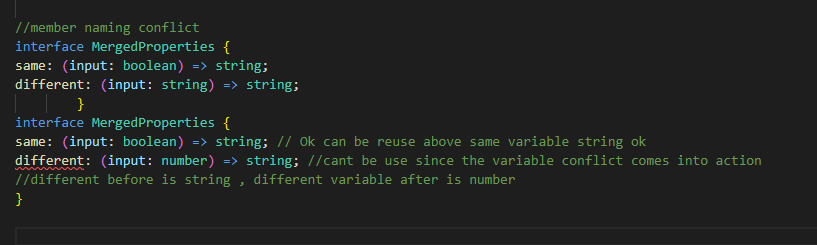
If two or more interface are declared with the same number of properties and with same name they can combine together as a single unit then

Text

Description automatically generated

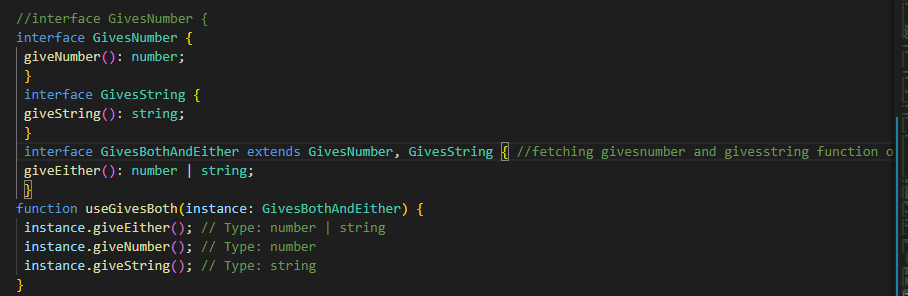
***Member naming conflicts:***

Data type variable could be confiict if type of defining a variable is different when wnna use it again.



***Extending multiple interfaces:***

We can use any function as a collab with any function , under other interfaces. By using extend keyword and comma with the extension in mind.



***Interface extension:***

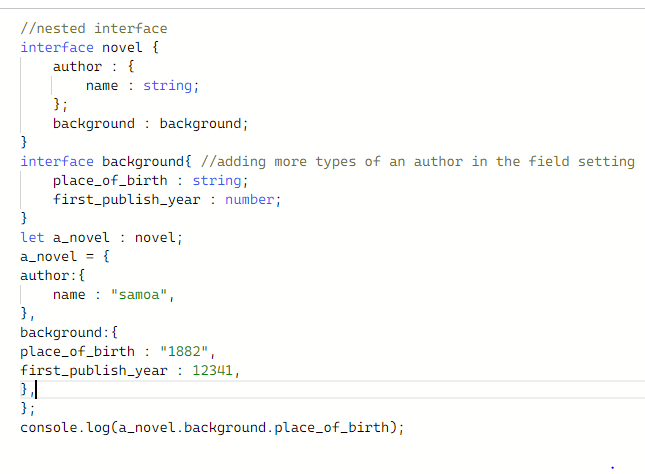
Under properties of interfaces thier interface types can also be nested along with other types in moreover interfaces. While this nested types can be called out while calling a first interface which directly dont preserve nested types but due to nested look at the second interface indirectly does.

Graphical user interface, text, application, email

Description automatically generated

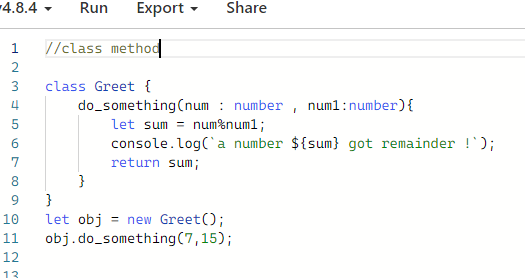
***Overridden properties:***

It is used to re declare property type of already present interface to enforce new values in it of different data types. By declaring another interface and extends old interface we wnna change its property type. Since if the more than one options types available we can reduce it through overridden option.

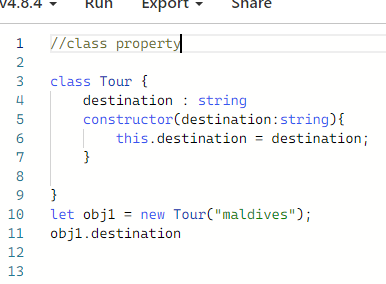


Chapter 8

***Class methods:***

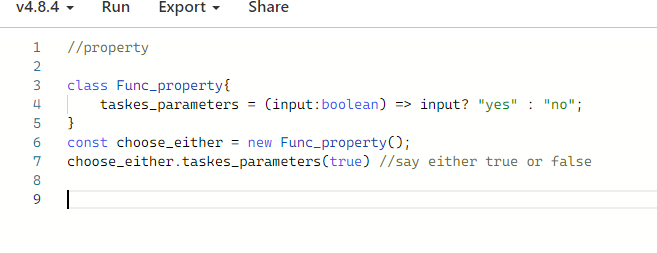


***Class property:***



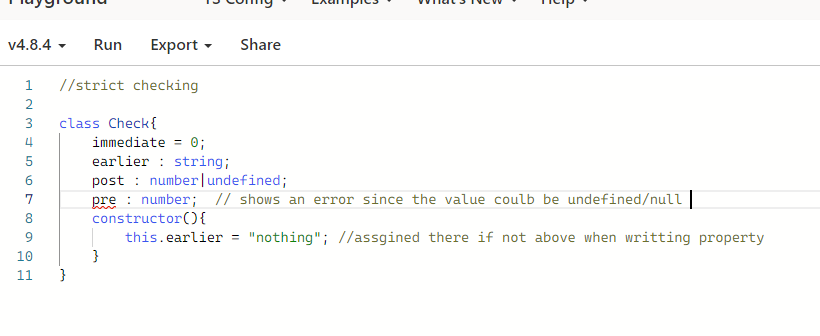
***Function property:***

a property could be an optional true false scenario we can say as instance. Which is mostly used as function property. Here look at this property option



***Strict initialization checker:***

It checks strictly if the defined property in the class is used in the constructor , method or not if not an error will be thrown to necessarily use this property out there.

This strictness while making properties prevents user from forgetting properties to assign values.

We can off this property through checkmark out

Graphical user interface, text

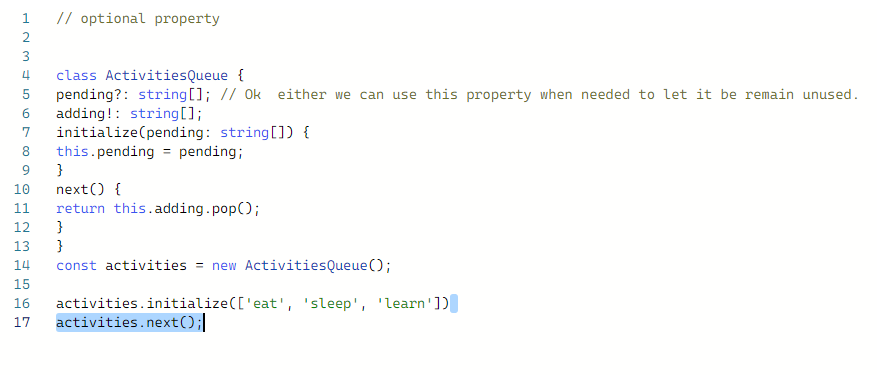
Description automatically generated

Example 02

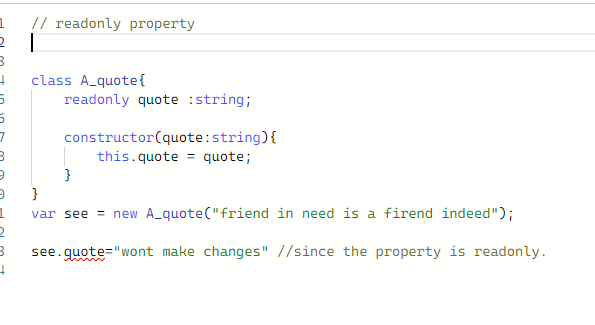
Graphical user interface, text, application, email

Description automatically generated

***Optional Property:***

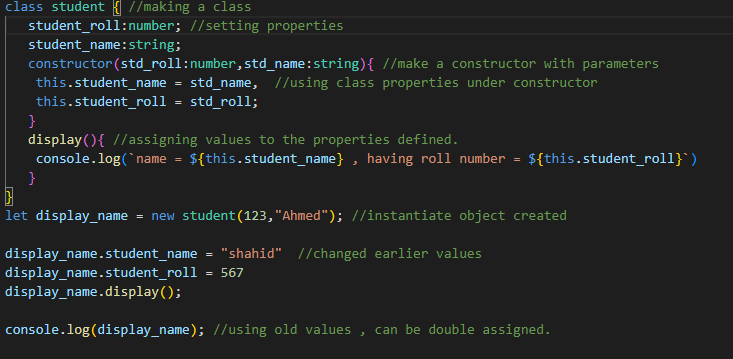


***Readonly Property:***



***Typescript class:***

using properties , instantiating , making constructor , parameters

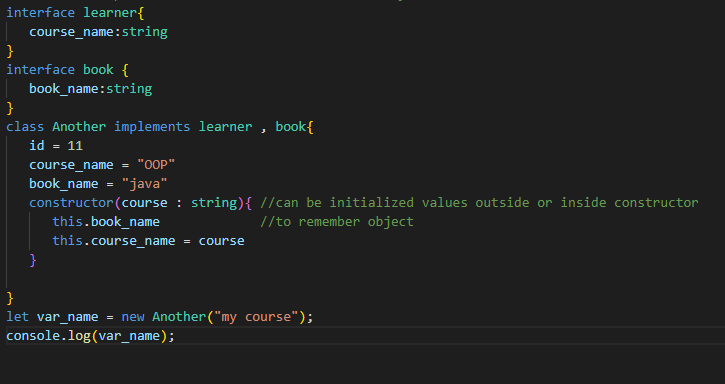


***Strict initialization checking:***

Text

Description automatically generated

***Interfaces + classes:***



Nominal type :

Type Pakistani contains by different infterfaces , objects , classes.

Structural type :