Resources: [TypeScript Tutorial #1 - Introduction & Setup (youtube.com)](https://www.youtube.com/watch?v=2pZmKW9-I_k&list=PL4cUxeGkcC9gUgr39Q_yD6v-bSyMwKPUI&ab_channel=NetNinja)

Use VSCode to run JS + TS

JS already built in VSCode

Need to install TypeScript

npm install -g typescript

Install extension:

Code Runner (more convenient to run only js)

Live Server (more convenient to run TS) -> then go to .html file -> right click -> open with live server

- ; at the end of line is optional for both, but JS will auto add ; while TS won’t

JS:

dynamic types, can change during writing

TS: (extend JS -> access anything JS has)

- strict type, CANNOT change type later, can change value (let, const)

- no int, float, only number type to cover everything

- hover to variable name, it will show the type

Browser don't understand ts, ts only makes code clearer, need to compile down to js

Command to compile it down to js

tsc sandbox.ts

tsc sandbox.ts -w

//-w: watch, auto compile to js, don't have to run again

// Will have error if open ts and js at the same time cuz same name

// Just close it, don't need to worry cuz browser cannot

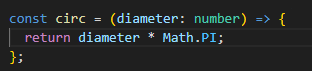
Browsers cannot interpret TypeScript code natively. So, it must be compiled to JavaScript before it can be executed.

<https://ggc.az1.qualtrics.com/jfe/form/SV_54QZ0R2oKkuU60u>

<https://github.com/iamshaunjp/typescript-tutorial>

3/ Type Basic

TS



Declare what type in parameter, in this case is number

Won’t compile to JS if wrong type is sent => type check, clean code, less error

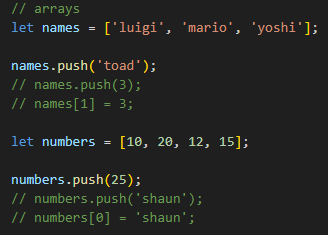
JS



No declaration

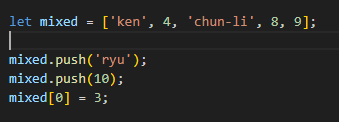
4/ Objects & Array

TS

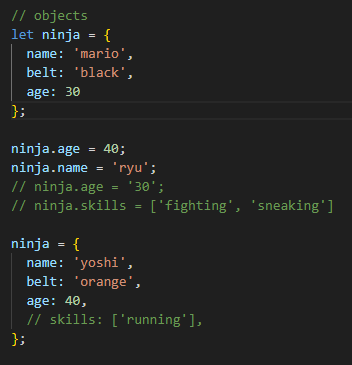


If you declare only 1 type, that’s the only one you can use

Cannot push/rewrite different type -> won’t compile to JS



If you want to mix, then just declare at the beginning (1st time)



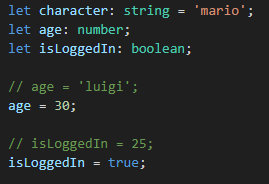
After declaring variable type, cannot change its type. But you can update whole object (with same props have been defined before).

Once we defined object, cannot add more properties to it (JS can add more props)

Once we declared an object, others have to be exactly the same as the initial one (value can be different)

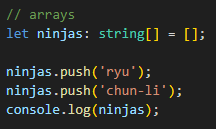
5/ Explicit Types

TS



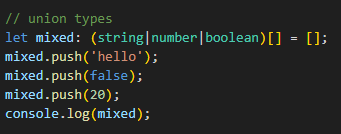
Initialize but don’t give value, only set the type for it.

Cannot set age = ‘luigi’ since we already give it explicitly a type (number)



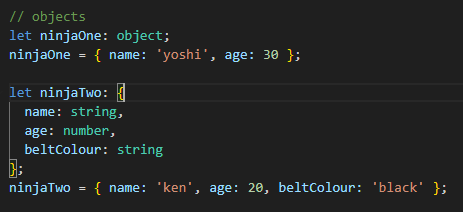
String[] = [] to initialize an empty array, so that we can use push. If we only have string[], and we use push, it will cause error

Array is also a kind of object



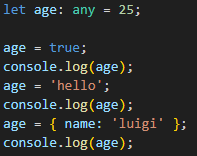
This array can be any of these 3 types.

 Union variable, uid can be string or number



Define an object or explicitly props for object

6/ Dynamics



Keyword “any” in order to use any type for variable, can change types later

This is reverse back to JS, it gets rid of TS benefits cuz it won’t show error during development about types, no hints of what types you will expected.

Only use when you don’t know the type, or need to change type later on.



Can use any for array and object too

But you might get things mess up since you didn’t declare the type for each variable (age & name were messed up)

7/ Better working flow

After split into folders

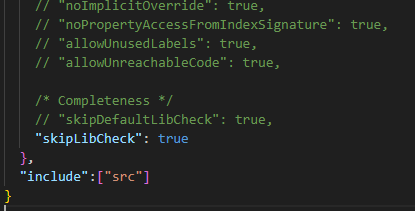
Public: .html, .js, .css

Src: .ts

Type in terminal tsc --init

It will create .json file -> open it

Change "rootDir": "./src" and "outDir": "./public"

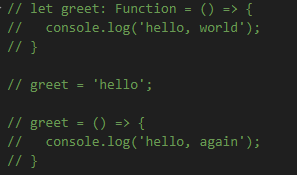


Add this so that it only tracks file in src. Files that are outside src won’t be convert to .js

Now go back to terminal type tsc -w

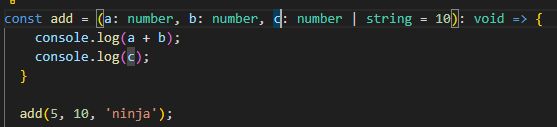
It automatically convert all .ts file in src to .js and put in public

8/ Function basic

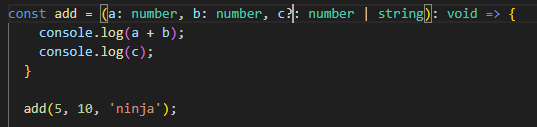


Already declare greet as a function, cannot change later on

Be careful Function is capital F



Set default value c = 10

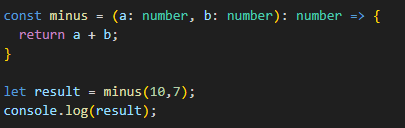


To set optional parameter, add ?

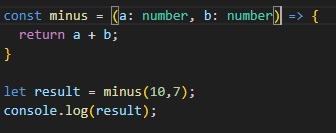
Don’t mix ? and default value, it will cause error, use either of them

Void: return value type (remove it is fine)

In js void = undefined but ts void <> undefined

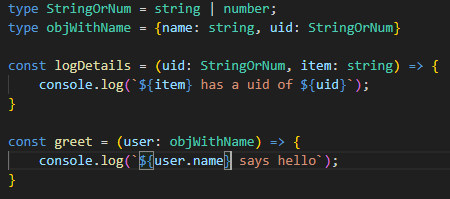


Return value type is number (if you want to explicitly define the type)



It’s ok if you don’t define cuz ts will automatically infer the return type

9/ Type Aliases

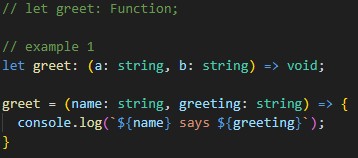


Create a type so that you can reuse later

${item} get value of item

${user.name} get name of user (since user is an obj)

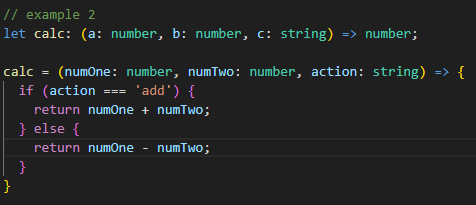
10/ Function signature



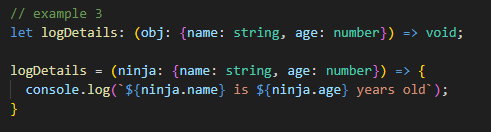
(a: string, b: string) parameters with specific type

Void: return value type

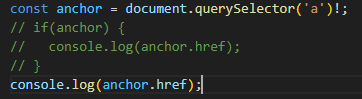
All types in parameter has to match with the 1st one that has been initialized



Function return a number

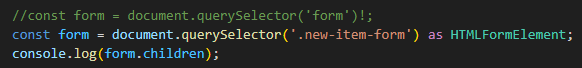


11/ DOM & Type Casting



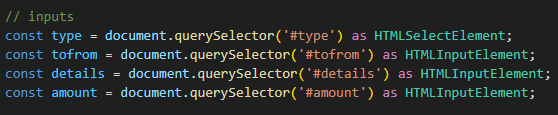
Need to check if anchor is null

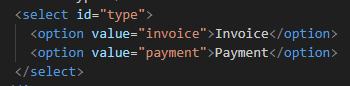
const anchor = document.querySelector('a')!; // The same as JS but TS add ! at the end (means I already known it’s not null)

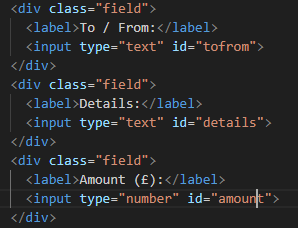


.new-item-form is from html form

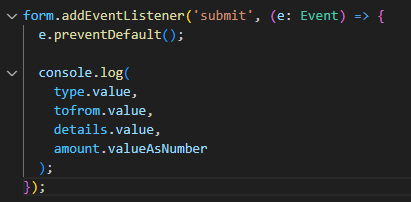
as HTMLFormElement : casting to HTMLFormElement, otherwise it only knows as Element







Get the id from html file, choose right HTML element (HTMLSelectElement, HTMLInputElement,..)



‘submit’ : event name

e: Event : parameter, Event obj

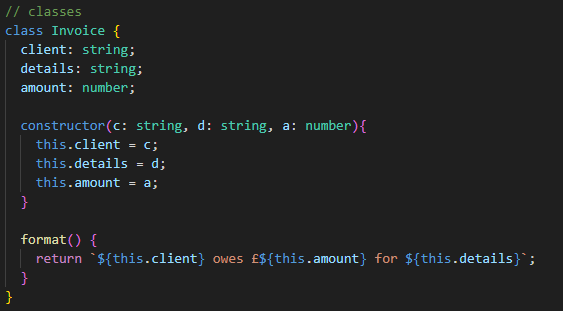
e.preventDefault() : prevent the page to refresh in default behavior when we submit the form

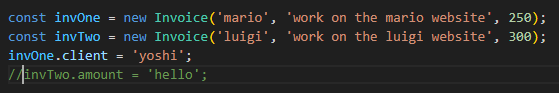
valueAsNumber : since JS auto turn value into a string, we need valueAsNumber to keep it as number

12/ Classes

In TS is very similar to JS

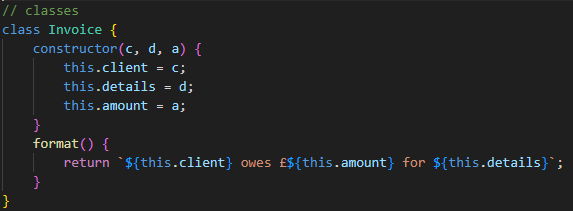
TS



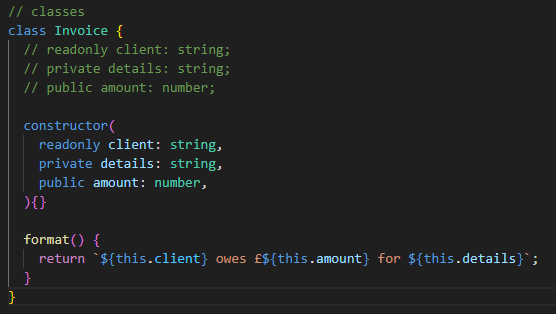


Everything is public by default so you can access and change value directly, but again you cannot change the type

JS



13/ Public, Private and Read-only



Access modifier (public, private, readonly)

public: can access & change directly from outside

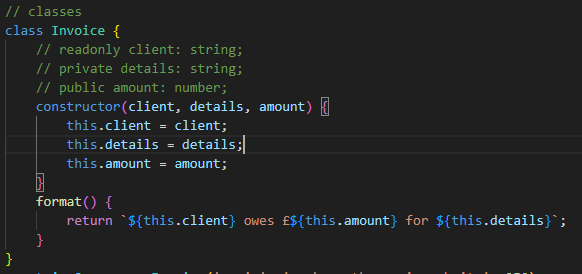
private: only access & change through methods in that class

readonly: can access from outside but cannot change value at all (inside + outside cuz it’s readonly), the property can only be assigned a value once, either in the constructor or at the point of declaration, and cannot be modified thereafter.

This is shortcut, we put all parameters into it, it will automatically assign the value, unlike JS

Need to put access modifier before prop name and type to make it work

JS

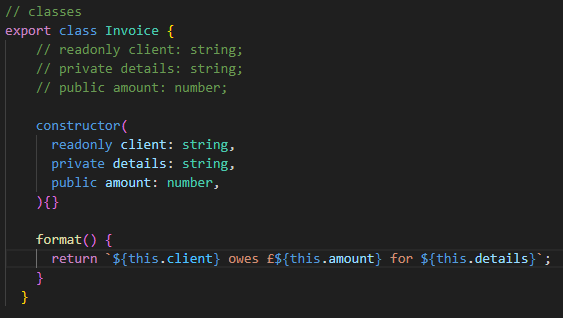


- Doesn’t have readonly, private, public -> needs to use closure (create a function and define var in that function) or naming convention (have the underscore \_ before var name, just for dev to know, not work)

- Have to do this.client = client… cuz no built-in like TS

14/ Modules

Go to tsconfig.json change "module": "ES2022", "target": "ES6"

Split files for better work flow. 



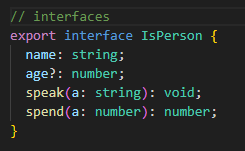
Go to app.js and invoices.js to check the import

15/ Interfaces

TS has interface to define the shape or structure of an object, specify the properties and methods that an object should have without providing an implementation. Interfaces are used to enforce a contract between different parts of your code, ensuring that objects adhere to a certain structure.

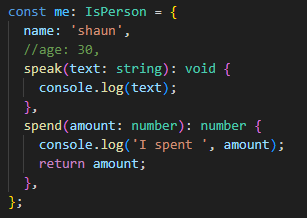
Still different from class, don’t use interface to generate, create obj, only use to enforce certain type of structure

JS doesn’t have interfaces



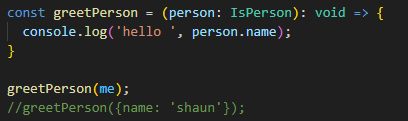
Don’t have constructor cuz we don’t use interface to create object

JS file doesn’t have this code block

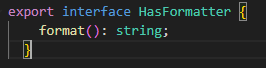


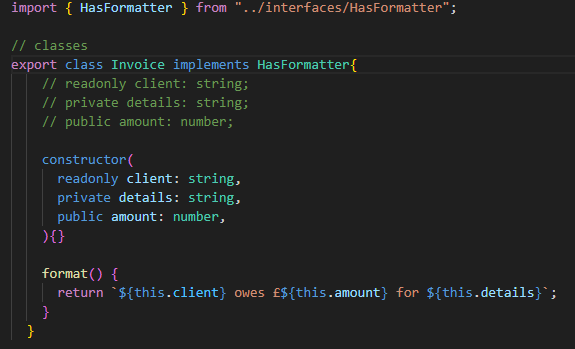
Create an object based on interface that we defined, need to match 100% the interface

Can have different obj with diff value, a bit diff methods but same signature, still have same props like interface, follow general structure of interface



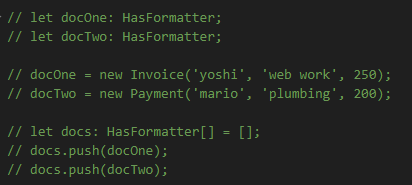
16/ Interfaces with classes





To make sure that this class invoice has a format method just like in interface, ensure that we have to follow structure

We have to have that format method, remove it will cause error

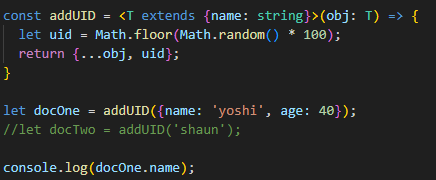


They both implement the interface so we can do this

17/ Render to HTML

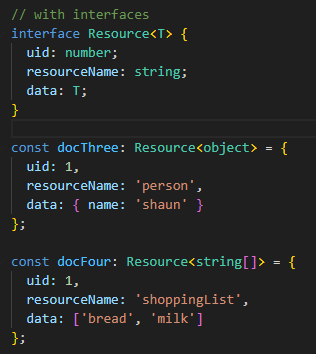
Need to create ListTemplate to export a render function

18/ Generics (Like generics in Java)



<T> generics

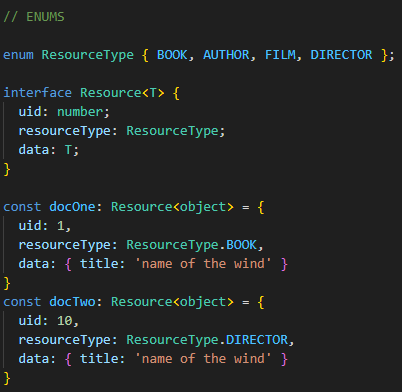
<T extends {name: string}> to specify an obj that has name: string



Pass the <T> in and let data be the T type

19/ Enums

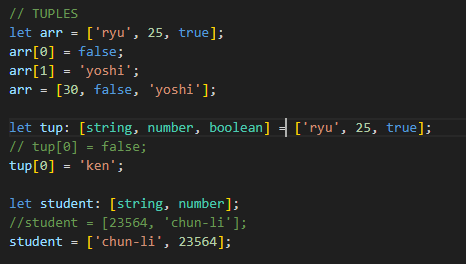
Associate number with a category



JS doesn’t have enum so it looks like this

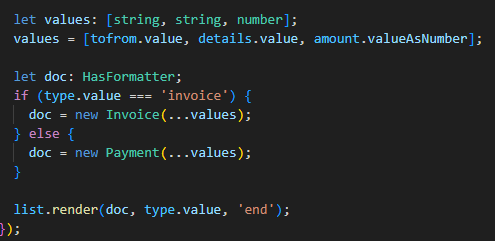


20/ Tuple



Strictly set the type for that position in the array, you can change value but cannot change the type

Now use in our code



We have to define the tuple in order for it to work (define all the type appropriately)

… is spread operator, get all parameter that passed in and put into values array