Angular Framework

What is angular

Used to develop client side applications for web, mobile, desktop and native applications.

It is mainly used to develop User Interfaces to create single page applications.

Single Page Application (SPA)

Everything happens in one page and any changes you do will update only part of the page instead of reloading the entire page,

SPA is much faster compare to multiple page applications, because SPA doesn’t need to pull the changes for the entire web page instead it has to pull content only for the part that needs to be updated.

Softwares required

Node.js - Runtime environment to run the angular applications

Editor - Visual Studio Code

Angular mainly uses two important technologies

* HTML
* Typescript

Typescript is a superset of Javascript it detects errors early and more reliable than Javascript

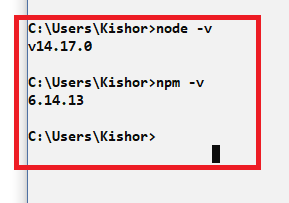
Inorder to develop angular applications, we need to install an Angular toolkit called as angular/cli

Angular CLI: It is a command line interface which provides commands using which you can create, run & build angular projects.

angular cli is downloaded from the internet which will have node modules which are javascript libraries, these node modules you can download only if you have node.js installed, node.js gives you one command called npm (Node Package Manager)

NPM: It is a tool to download any javascript libraries including Angular/CLI, React Toolkit.

Verifying node & npm



You can install angular/cli using the following command

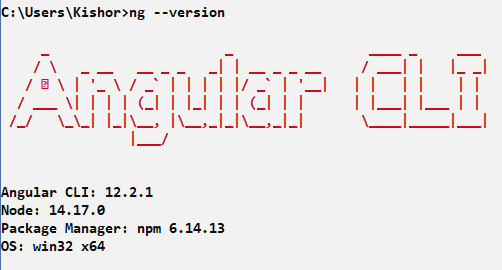
* npm install @angular/cli
* npm install -g @angular/cli

npm install @angular/cli: It installs angular toolkit locally, in the location your terminal is opened, you can create angular projects or run angular projects only in that location

npm install -g @angular/cli: It installs angular toolkit globally, you can create/run/build angular projects in any location

ng: this is command you will get once you install angular/cli, this command allows you to create angular projects, run and build the angular projects

Verifying the angular cli



Some of the useful commands in angular through ng

ng new app-name: it is used to create a new angular application, where app-name is the project name

ng serve: it is used to host your angular application & runs in a default server provided by angular in port 4200

Other than this you have commands like

ng generate component component-name (or) ng g c component-name

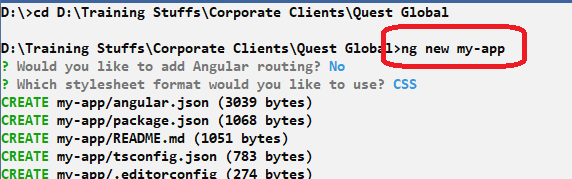
ng generate service service-name (or) ng g s service-name

ng build

ng test and so on.

Creating our first angular project

ng new my-app



After installation is complete, you will get a folder which is an angular project, you need to navigate inside the project from the terminal so that you can run your angular application or create angular programs.

What will the angular project have?

The angular project will have many features

* It will have auto-compilation feature which automatically compiler your angular application when you update the changes
* It will have an embedded server to host the angular application
* It will have a live-reload feature, which shows the output when you do changes in the application
* It will have all the supported Javascript libraries (node modules) required to develop angular application

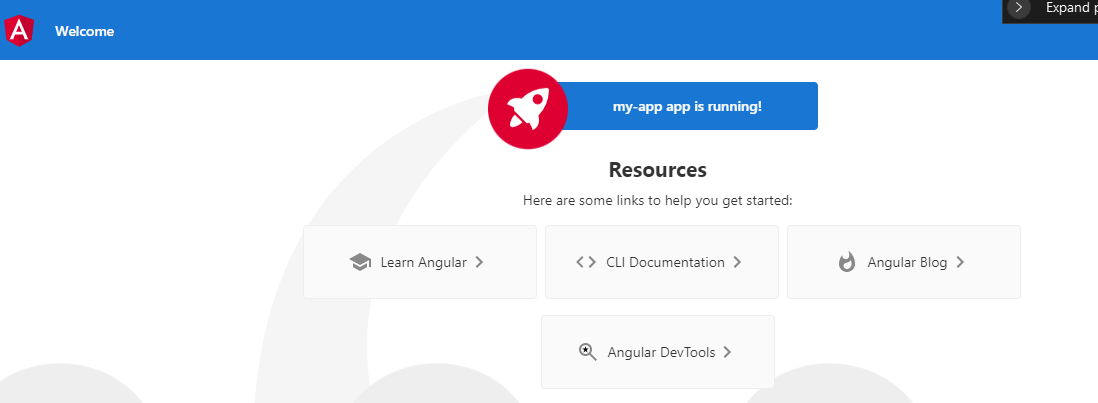
Note: Once you create the angular project using ‘ng new’ you will get a ready to run project with some default content

You can verify the project download completing by launching the application using ng serve

Note: *ng serve* should be entered from the project location i.e., parent directory, in my case it is my-app

Once you enter ng serve the command prompt shows compile successfully and angular application is running in port 4200, you can test the application is running in browser by entering <http://localhost:4200>

Output:



The above output is common for every new project which we need to change as per our requirement.

Summary:

* Verified node & npm using node -v & npm -v
* Installed angular/cli using npm install -g @angular/cli
* Created a new project ng new my-app

Note: Installing angular/cli is one time job

Angular uses two languages mainly

1. HTML - used to develop content for web pages
2. Typescript - used to write application code, which is a super set of Javascript

Javascript: It is a program written for web pages to make your web page more interactive, Angular uses Typescript which is a super set of javascript that is compiled by angular to convert to the javascript

Note: Angular converts Typescript to Javascript so that every browser can understand

Typescript make use of lot of new features of Javascript i.e, ES6 features

ES6 is also known as ECMAScript2015, released in 2015, ES6 is a standard which provides some rules that is implemented by Javascript & all the browsers.

ES6 has provided some features to improve the syntax of the Javascript to easily write the program, earlier Javascript used ES5 feature whose syntax were bit hard to write & understand.

Some of the features of ES6

* Declaring variables using let & const keywords
* Introduction of classes, extends, super keywords to make Javascript object oriented
* Rest & Spread operators
* Default function parameters
* Object Destructuring
* Arrow Functions
* Template string literals

Browsers understand 3 technologies

* HTML (.html)
* CSS (.css)
* Javascript (.js)

You can include css & javascript into HTML and open HTML in the browser to see the output

Purpose of let & const keywords

These are used to create block scoped variables in Javascript, earlier before ES6 javascript variables were global it means it can be accessed anywhere in the program even if the variables are created inside the loops or conditions or functions.

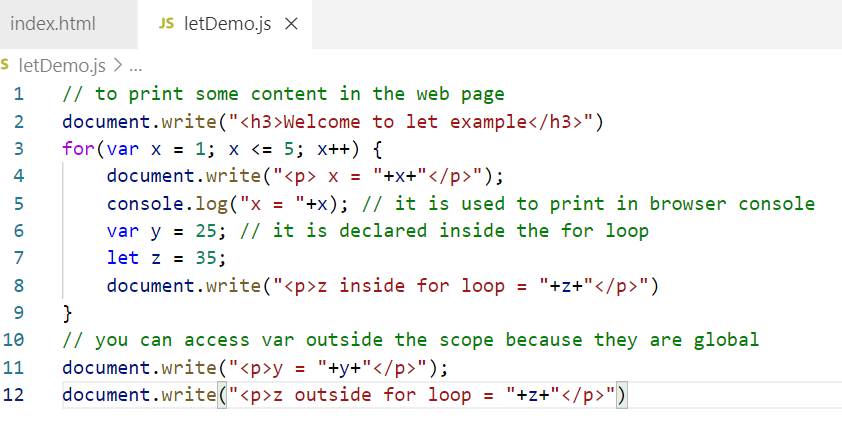
How earlier javascript variables were created

Using var keyword

var username = “Alex”;

var age = 35;

letDemo.js



index.html



Output:

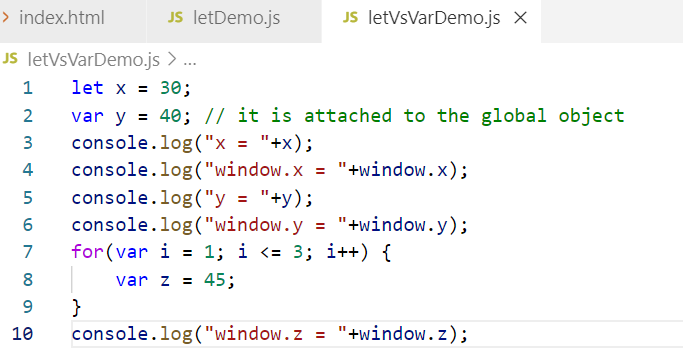


You can notice that let variable is not accessible outside the scope, the reason var variables are accessible is because it is attached to the global object of the program i.e., window

You can access the var variables using window object also, because these are added as a property of global object

ie.., window.y

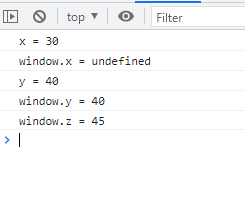
letVsVarDemo.js



index.html



Output:



the var is attaching the property into the window object hence they become global, but let variables are not added to the window object.

const keyword

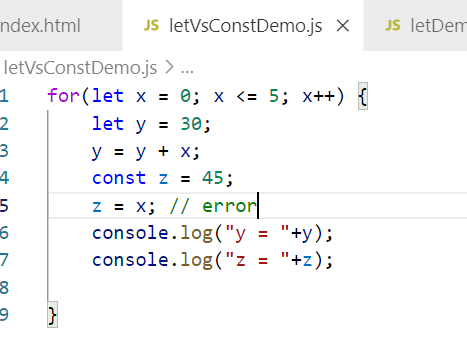
const is another block scope variable you can create, but its value can’t be modified, it is read-only once declared

const PI = 3.14

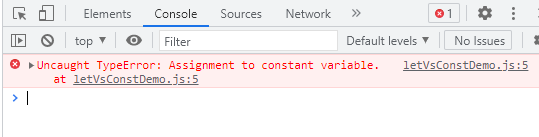
PI = 4.4; // error

let x = 30;

x = 45; // OK



Output:



var Vs let Vs const

var will become global and you can re-declare the var variables

i.e.,

var x = 30;

var x = 40;

let & const are block scoped variables, let can be modified however const can’t be modified, re-declaration is not allows in both let & const

let x = 40;

x = 15; // OK

let x = 50; // error

const y = 10;

y = 5; // error, trying to modify the const

const y = 10; // error

Using const for Javascript objects

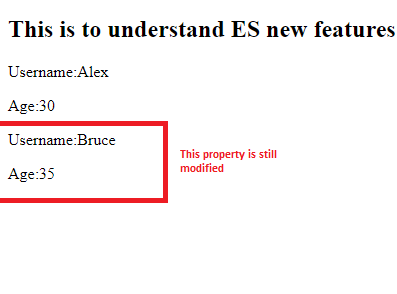
const emp = {name : “Alex”, age : 35, salary : 40000};

emp = {name: “Bruce”, age : 40, salary : 45000} // error

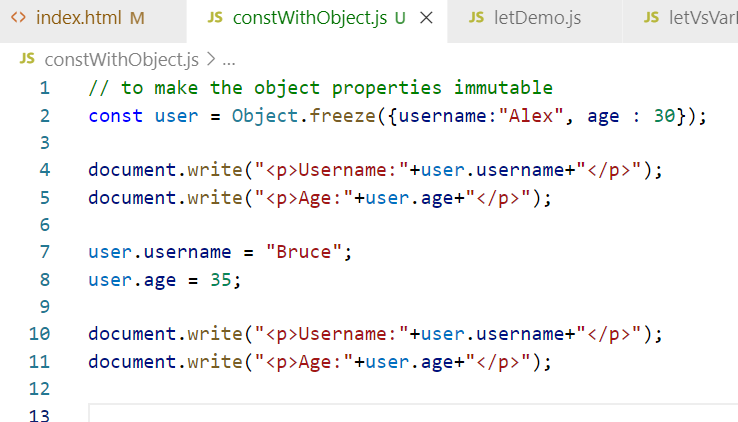
The const keyword restricts modification, however you can still change the value of the object.



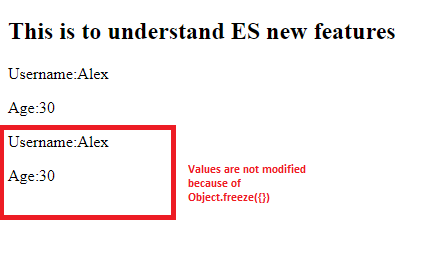
Output:



You need to use Object.freeze({}) to make the object properties immutable



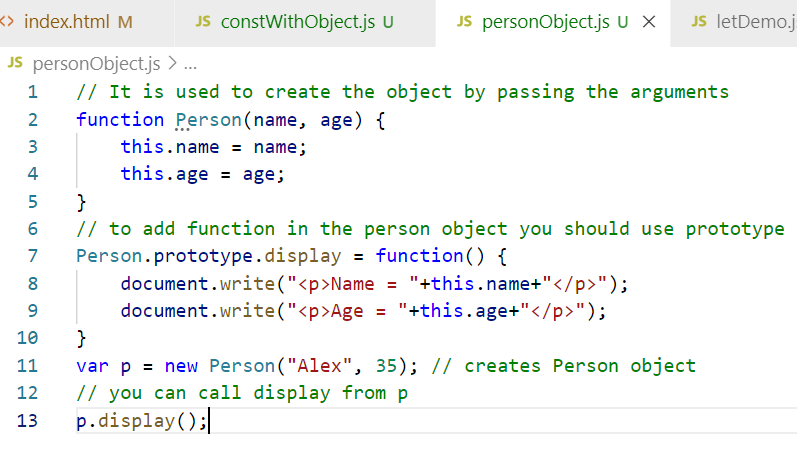
Output:



How to add methods to the object

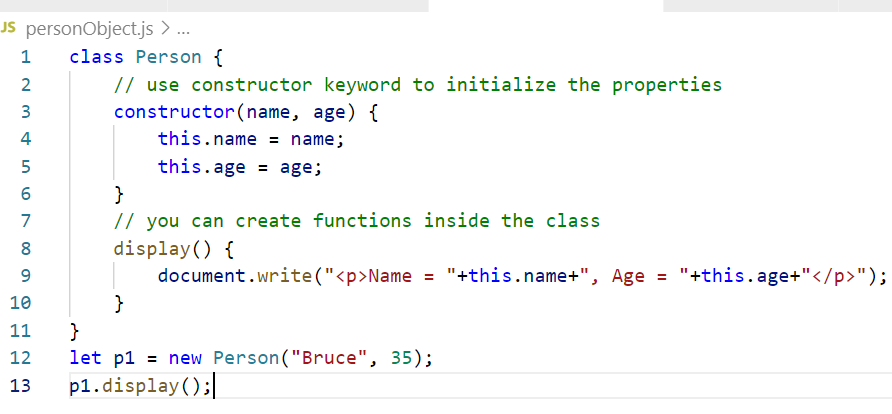
Earlier you need to use prototype property to add the methods to the javascript object

Before ES6 you need to use prototype to create function for the object



But from ES6 onwards you can use class keyword to create the constructor & function inside the class.

From ES6 onwards you can create class as below



Inheritance: Process of acquiring the properties & behaviour of an object from another object

Before ES6 you need to use prototype to achieve inheritance

function Person(name, age) { }   
Person.prototype.display = function() { }

function Employee(id, name, age) { }

If You want display() function in employee then you need to inherit using prototype as below

Employee.prototype = Object.create(Person.prototype)

e = new Employee(1, “Raj”, 35);

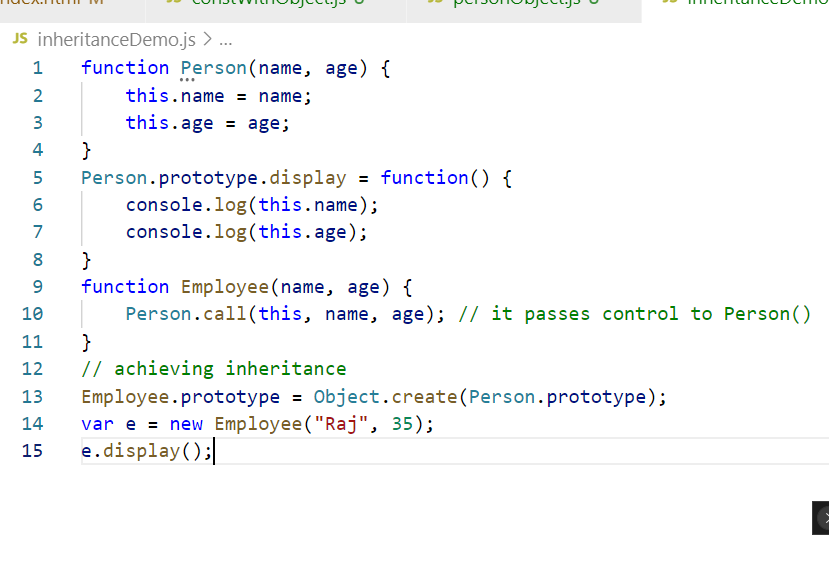
e.display();

From ES6 onwards you can use extends keywords

class Person { … }

class Employee extends Person { }

Old approach of inheriting the functions



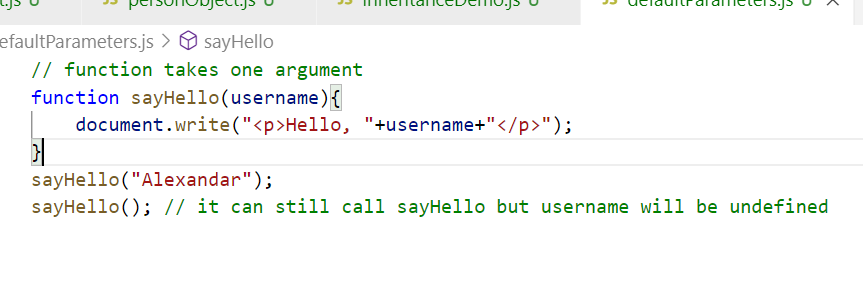
New approach of inheriting the functions



Default parameters to the function

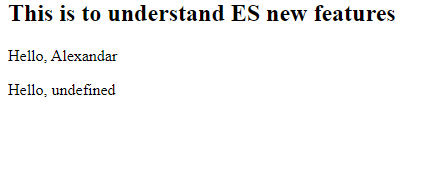
In Javascript you can call the function having arguments without passing right number of arguments, in that case you don’t get error instead the missing parameters will be undefined

Ex:

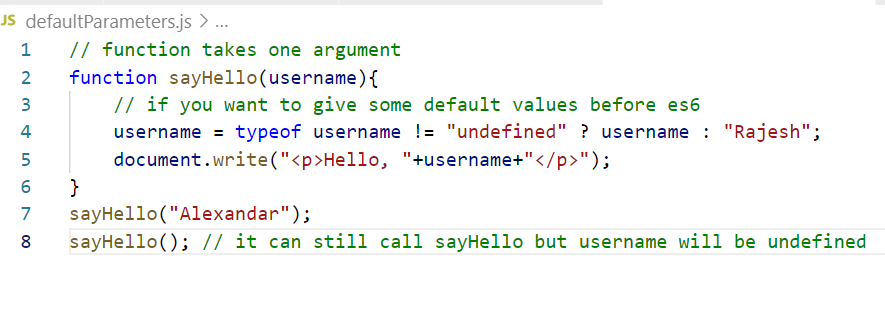


Here you are calling sayHello() without passing parameter in that case username will be undefined.

Output:

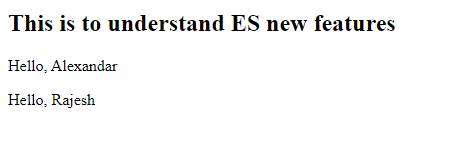


Earlier to ES6 developers need to write some logics to avoid undefined



The above code checks the type of username is undefined or not, if yes then assigns the default value Rajesh, if not then assigns the value you passed as the argument.

Output:



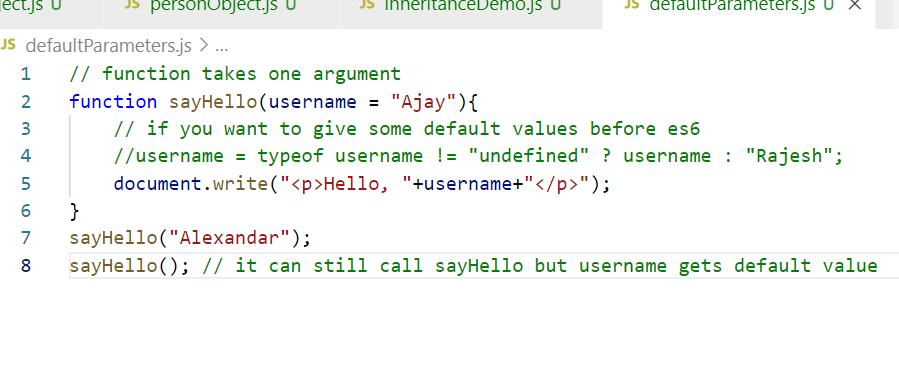
But in ES6 its much easier to provide the default values, i.e., in the parameters itself you can assign the value to the parameter so if the argument is passed then the parameter takes the value passed in the argument, else the parameter takes the default value.

function sayHello(username = “Ajay”) // default value is Ajay  
{   
}

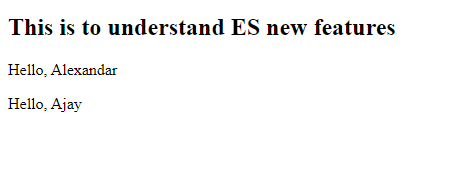
sayHello(“Kiran”); username will be Kiran

sayHello(); username will be Ajay

Default parameters in ES6 is much easier as below

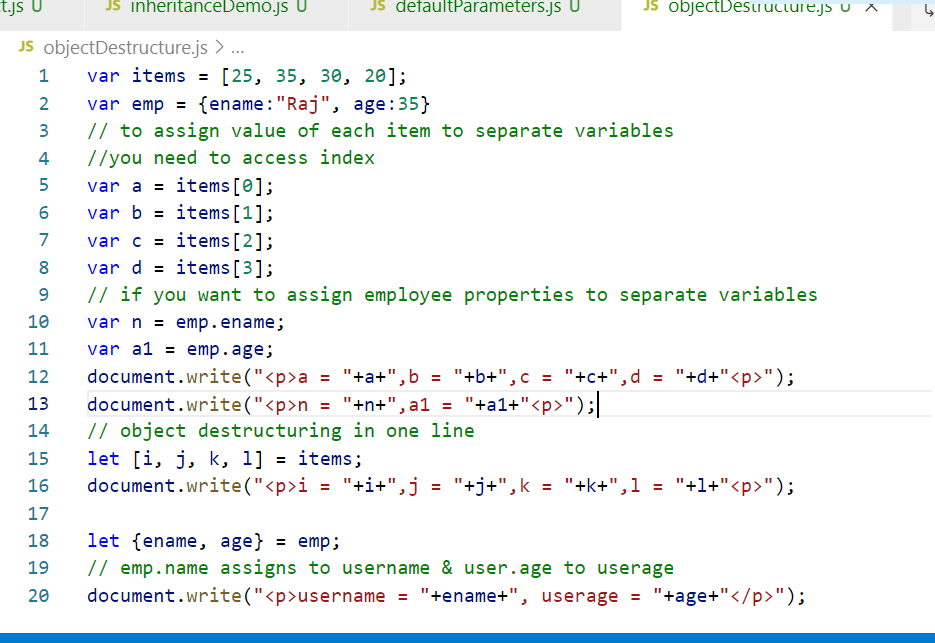


Output:



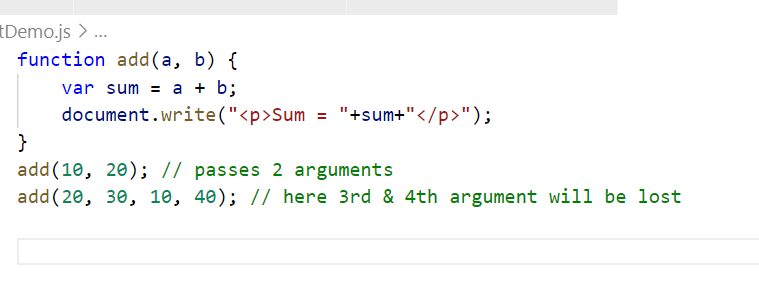
Object Destructuring

It allows you to assign values to multiple variables without accessing each index of a complex object



Rest Operators

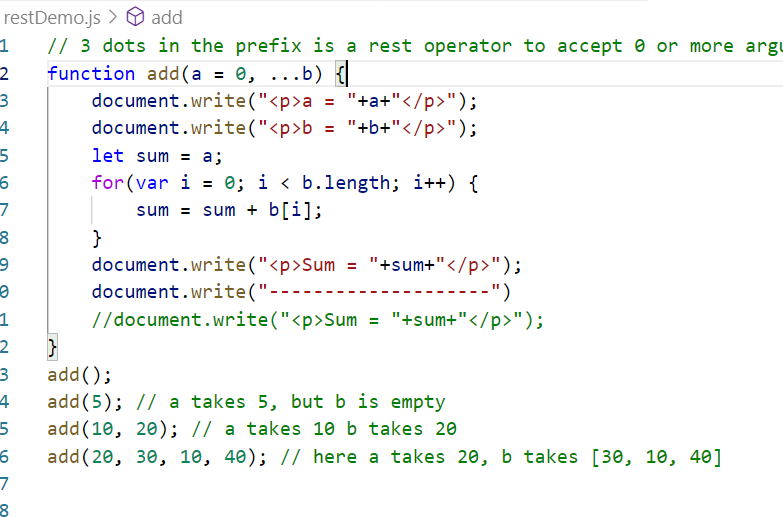
Before ES6 a function can only take specified number of arguments based on the number of parameters



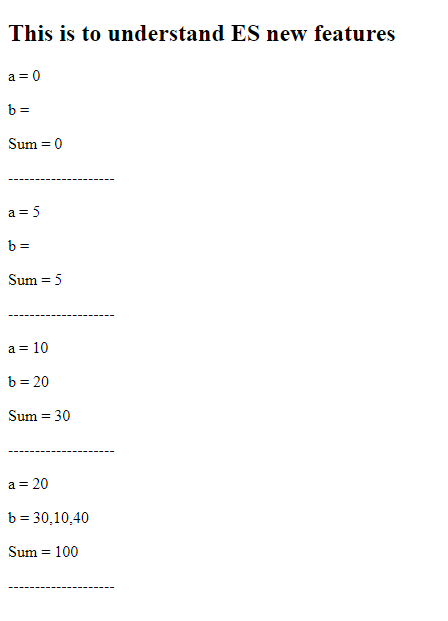
In ES6 you have a rest operator where a variable can accept 0 or more arguments like an array

With Rest operator you can avoid losing the data

restDemo.js



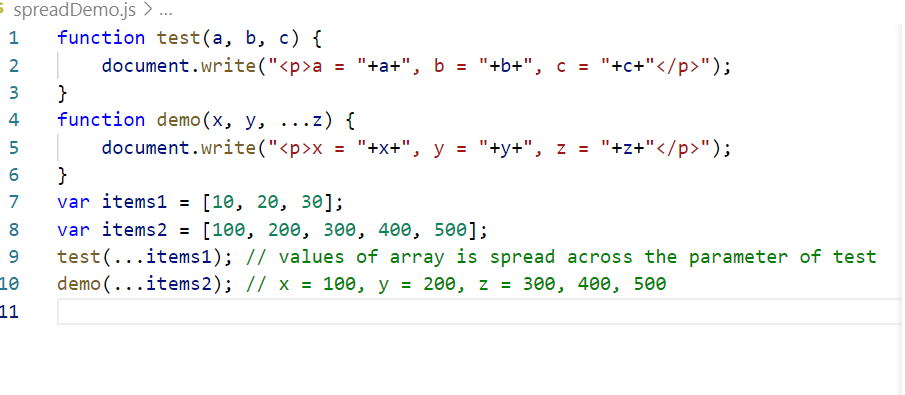
Output:



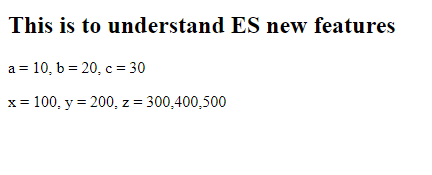
Spread Operator

It is used to spread the arguments to multiple parameters of a function

spreadDemo.js



Output:



Other features of ES6 are

* Template literals
* Arrow functions

Some other features introduced in ES7, ES8, ES9 and etc

* Exponential Operator: to simplify writing exponent expression without using Math.pow()
* includes() in array: to find the content of an array exists or not
* Object.entries() & Object.values() to easily access the object properties
* padStart() & padEnd(): to add some characters repeatedly in the string at the begging & the end
* Trailing commas: to avoid errors when developer forgets to write any properties in the object
* Optional Chaining: to safely access the object properties without any errors

Template literals: It is used to access the data in a string without breaking it

Earlier: You need to concatenate the data to the string using + operator

ex: url = <http://domain.com/employees/1/dept/102>/

If 1 & 102 are id & department id, you need to access it at the runtime and concatenate in the string as below:

empId = 1 // assume it is read dynamically

deptId = 102 // assume it is read dynamically.

url = “http://domain.com/employees/”+empId+”/dept/”+deptId+”/”

Now the url will be = <http://domain.com/employees/1/dept/102>/

With the help of template string literals you can create a string without using + operator and you don’t need to break the string, you need to use back tick(`) to create string.

url = `http://domain.com/employees/${empId}/dept/${deptId}`

Here ${empId} will be replaced by 1 & ${deptId} will be replaced 102 the above url will be same as below:

<http://domain.com/employees/1/dept/102/>

This also helps in adding HTML tags with dynamic content without breaking the string.

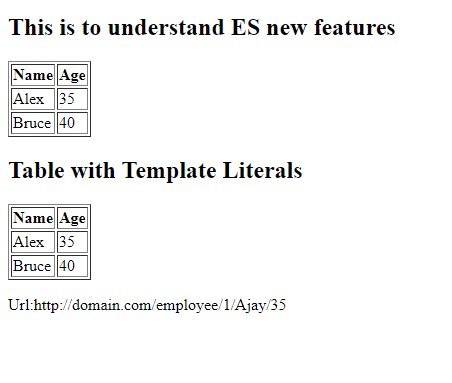
Earlier:

If you have array of some users in some variable like:

users = [{name:”Alex”, age:30}, {name:”Bruce”, age:40}];



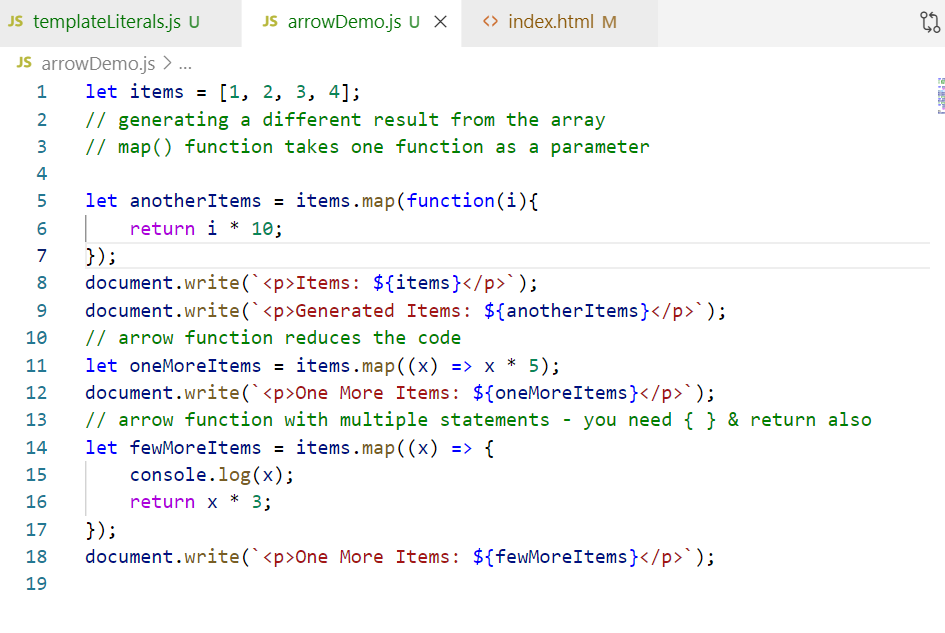
Output:



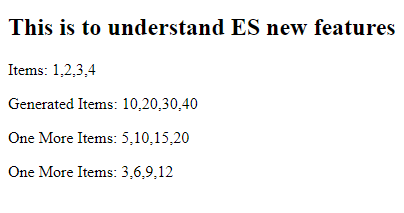
Arrow function

It is a simplified form of writing the anonymous function or callback function

arrowDemo.js



Output:



Arrow functions doesn’t need to specify {} if you it is one line statement & also return keyword is optional.

If the function is taking only one parameter then you can also avoid writing () parenthesis

i.e, items.map((i) => i \* 10) can be written as items.map(i => i \* 10).

Exponential Operator

You can use \*\* to perform exponential operation without using Math.pow() function

Earlier: Math.pow(2, 3) // 8

From ES7 onwards: 2 \*\* 3 // 8

Includes in array

names = [“alex”, “bruce”, “Charles”];

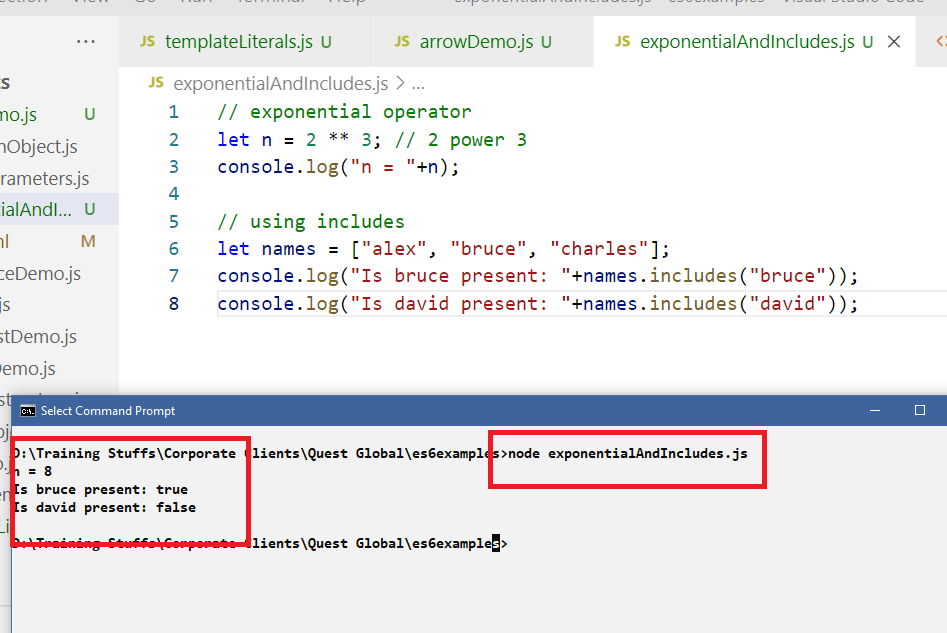
You can check a particular value is present using includes method

names.includes(“bruce”); // true

names.includes(“sachin”); // false

You can run javascript not only in browser, you can use node.js command to run i.e., node file.js

exponentialAndIncludes.js

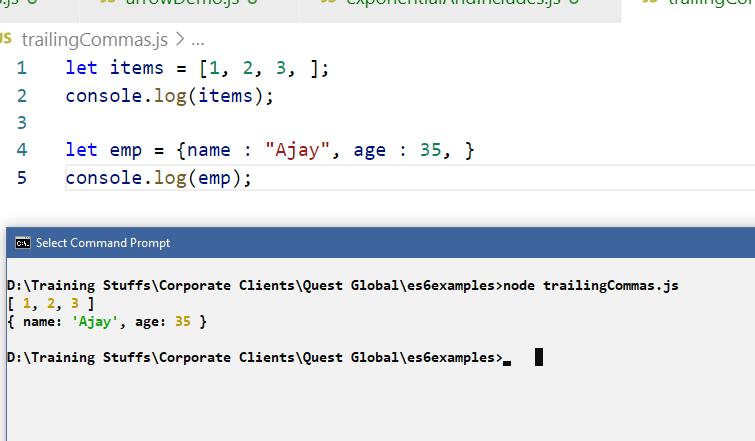


Trailing commas

With this feature, the commas you write after some elements in the array or some properties in the object are ignored, earlier javascript was throwing error, now it will be ignored

items = [1, 2, 3, ] // earlier it was an error but now it’s okay

emp = {name : “ajay”, age : 35, } // now the commas are trailed



padStart() & padEnd() functions in the string

Earlier developers were writing too much code to add some extra characters in the beginning or end of the string, but with padStart() & padEnd() it is simple.

Ex: Assume you want to show otp of 6 digits, but random number generated 5 digits like 78235, then you need to add 0 in the beginning and show 078235, if it generates 4 digits like 7234, then you need add two 0’s in the beginning, 007234

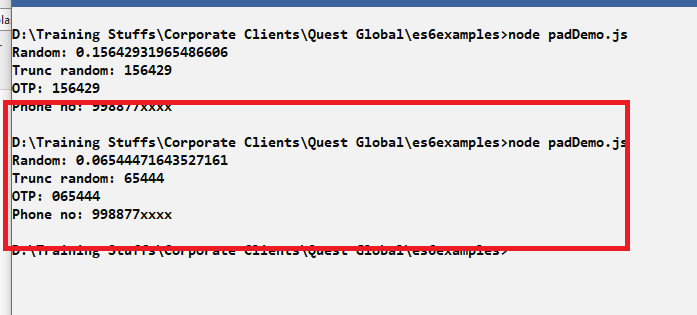
Ex: Assume you want to show phone numbers where last 4 digits with some character instead of actual number i.e., 987135xxxx, here you need to add x 4 times to the 10 digit number, for that you need to extract first 6 digits and add x 4 characters to the end, but with padEnd() you can do it much easily

padStart(): adds the character in the beginning till the end you specify

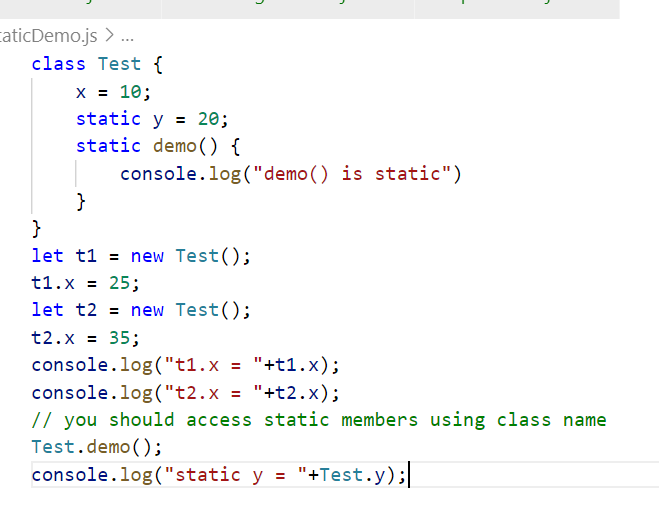
padEnd(): repeatedly adds the character from the end of the string.



Output:

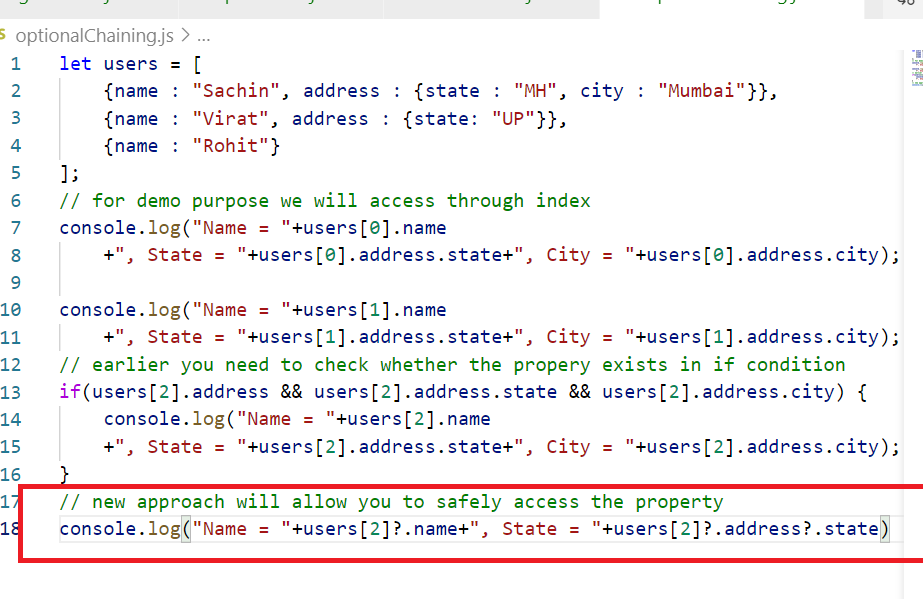


static keyword: It is used to access the member without creating the object and also you want to share fixed configuration across multiple objects

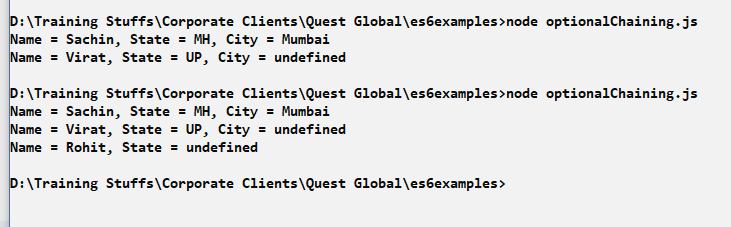


Optional Chaining(?.): It is used to safely access the property without any error.

It avoids developers to constantly check for properties or nested properties in the object



Output:



Summary of ECMAScript or ES

* let, const, class, super, extends, constructor keywords
* Default functional parameters
* Object Destructuring
* Rest & Spread operators
* Arrow function
* Template string literals
* padStart() & padEnd()
* Exponential Operator
* Optional Chaining
* Array Includes
* Static
* Trailing Commas.

Typescript

It is a programming language which is a super set to the Javascript, super set means, it supports all the features of Javascript in addition to that it has some extra features.

Typescript is more predictable & reliable when comes to expecting the result, because adds type annotations to the Javascript and typescript will be compiled first which will be converted to Javascript, it catches the error at the early stage and you can avoid lot of runtime errors which was a major issue in Javascript.

Note: Typescript is not directly executed on either browser or on any Javascript runtime environment, you will execute the compiled Javascript only

Typescript provides types to the variables, function parameters & return types along with that it allows you to write code much easily compare to Javascript

Problems with Javascript

* A variable can take any kind of value, the output would be unpredictable
* A function can return any kind of value
* Importing the function in another script is a complex syntax
* More of errors when the operations are done with some unexpected values
* Code is not compiled for early detection of errors.

Ex: function add(x, y) { } // in Javascript you can call add() with any arguments, with any type of value also

add(10, 20); //ok

add(“hello”, “hi”); //ok

add() // ok

In Typescript all these things you can avoid, because it adds type annotations

function add(x : number, y : number) {   
  
}

Now add can be called only by passing 2 argument those must be number type

add(20, 10); // ok

add(“hello”, “hi”) // error

add() // error

Typescripts are written in .ts file, which will be compiled to .js file, we need to run the .js file not .ts file

Inorder to compile typescript we need typescript compiler

Note: In Angular typescript compiler is embedded in the project you don’t have to install, but if you want to manually write typescript then you need to install typescript compiler

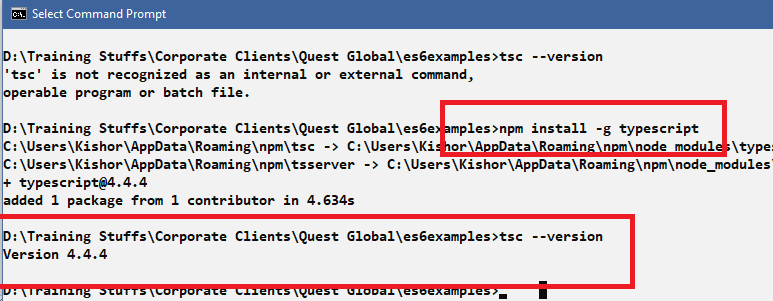
How to install typescript

>> npm install -g typescript

How to verify the typescript installation

>> tsc --version

tsc means typescript compiler



Types in typescript

* number: for numbers
* string: for strings
* boolean: for boolean values
* any: a variable that can take various types
* void: absence of value
* []: for array

How to compile typescript code

tsc file.ts

This generates the javascript that is compatible with ES5, to make it compatible with all the javascript runtime environment

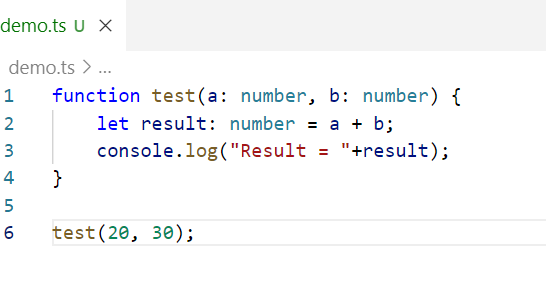
You can compile typescript to generate the javascript with new features

tsc --target es6 file.ts >> generates the javascript compatible with es6

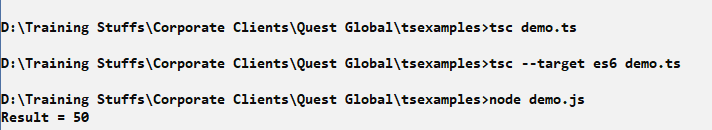
tsc --target es2015 file.ts >> same as es6

tsc --target es2020 file.ts >> compatible to ES9

demo.ts

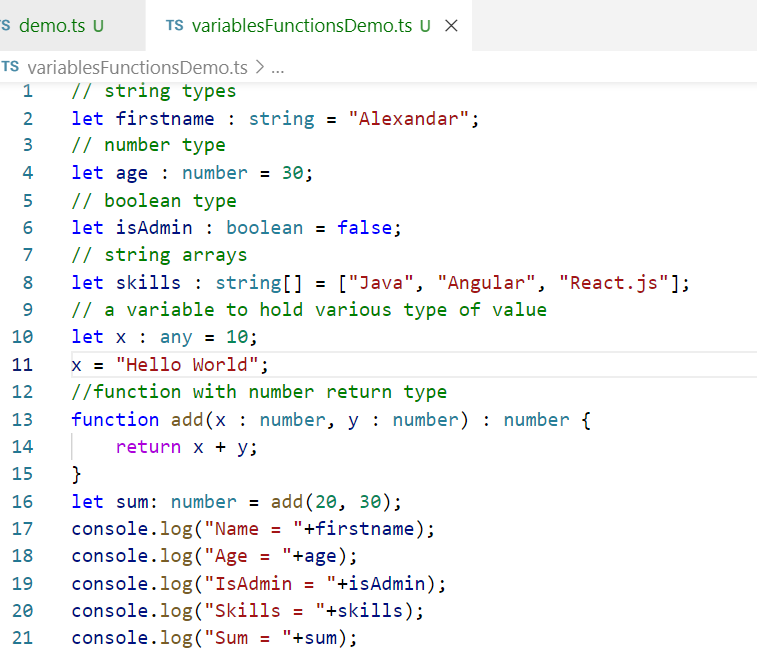


Output:

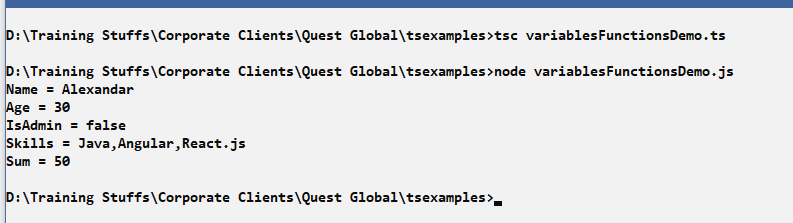


Creating variables with different types

variablesFunctionsDemo.ts



Output:



Tuples in typescript

It is used to represent arrays but with fixed set of elements with types, the type need not be same for each elements.

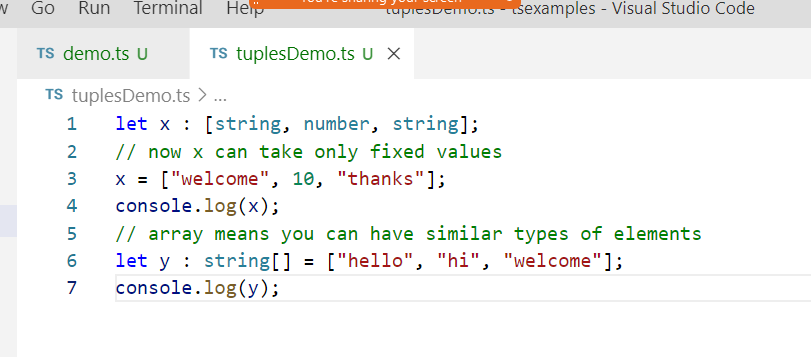
Syntax: let x : [string, number, string]

Here x can take values of types in the order you have declared i.e., x = [“hello”, 10, “world”]

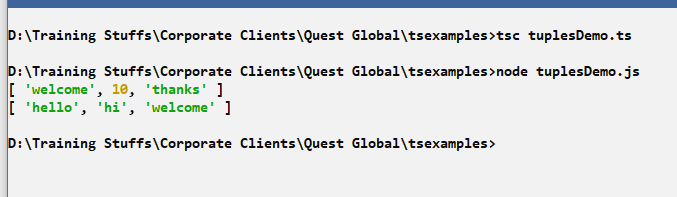
But x = [10, “hello”, “world”] // error

In case of array you can only mention defined types of values, for ex: x : string[], now x can only take strings.

tuplesDemo.ts



Output:



Union types

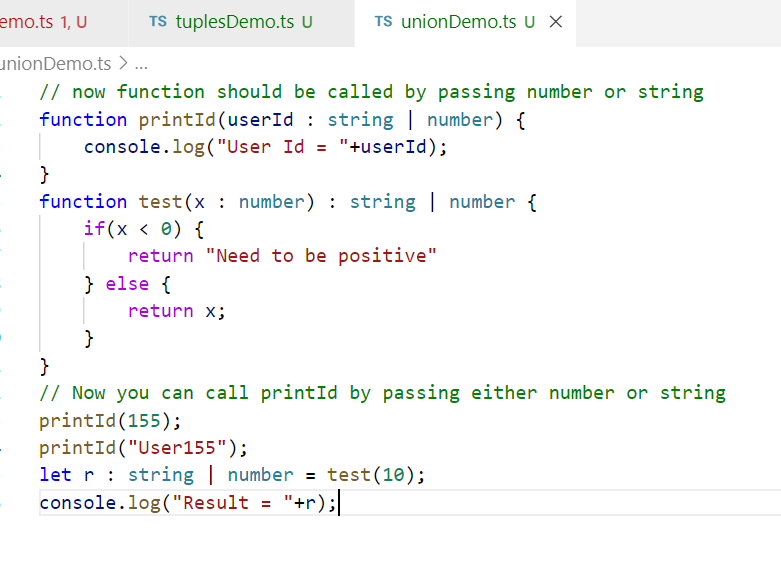
It is used when two or more types to be represented

Syntax:

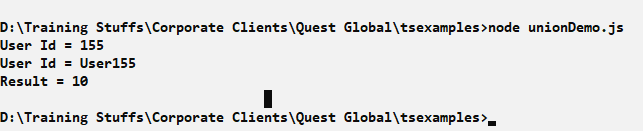
variable : type1 | type2 | type3 and so on

Now the variable can take only the types defined

unionDemo.ts

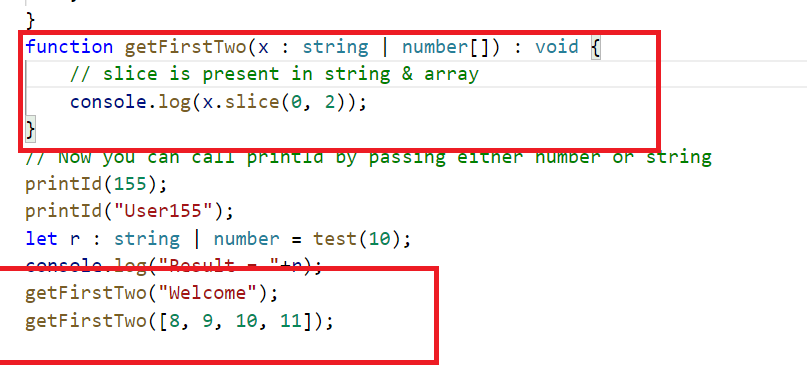


Output:

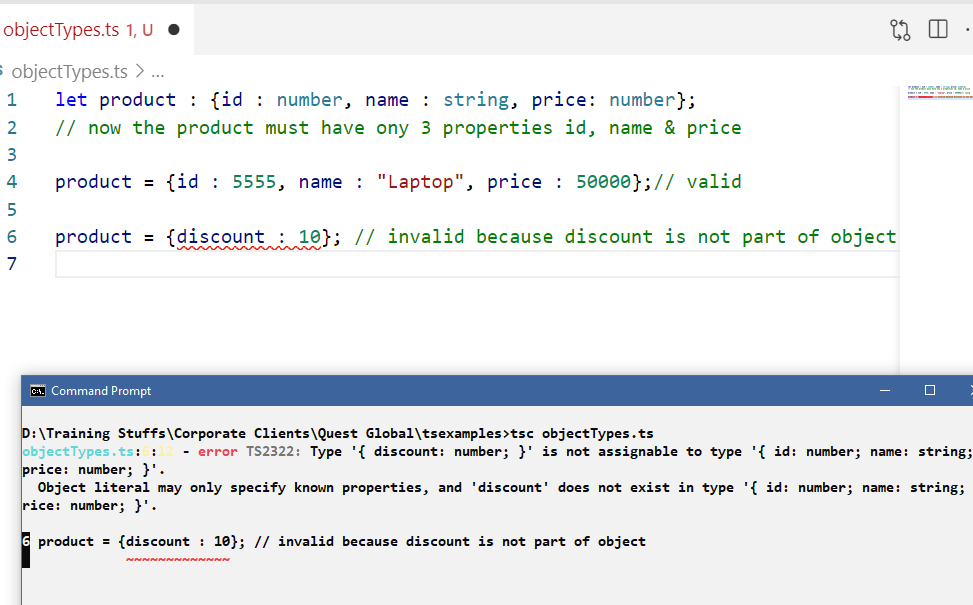


Note: when a union type is string | number, you may not able to access substring() because its available only in string not in number, but in some scenarios the union types may have something common in all the members in that case it wouldn’t be problem to access that common property/function.

ex: array & string both has slice, so if they are used in union you can call slice



You can also mention types in the object properties so that you can expect the correct values



Variables & Functions with optional properties & parameters

Sometimes you may have an object with optional properties which is not mandatory, like lastname, phoneNumber may not be mandatory in that case you can use optional properties using ‘?’

user : {firstname : string, lastname?: string, phone?: number}

Here user object must have firstname, but lastname & phone are optionals

You can also create optional parameters, where the arguments are not mandatory

Note: It is not similar to default or rest operator

function test(x: number, y?: number) {

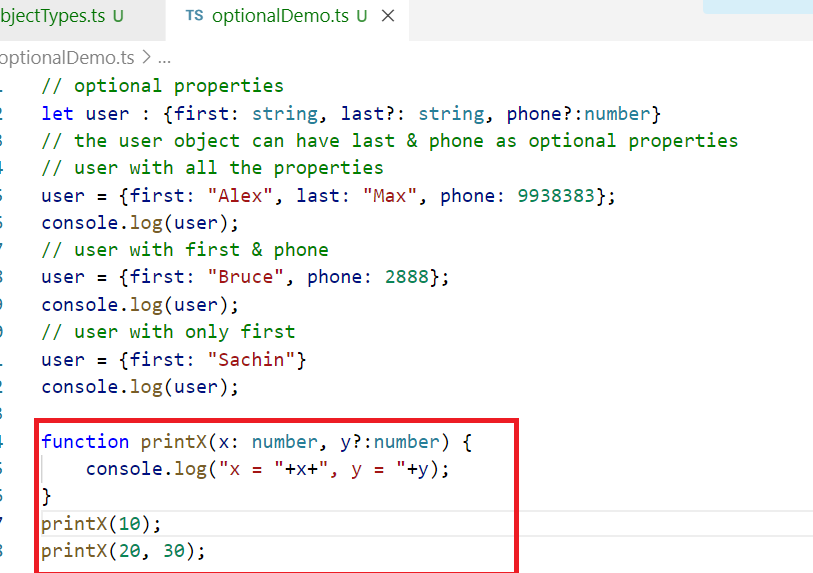
}

Here the y is optional

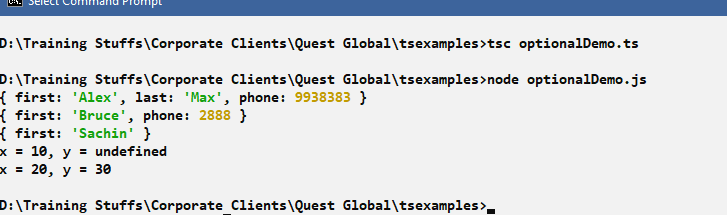
test(10); // ok

test(10, 20); // ok

optionalDemo.ts



Output:



Access Modifiers

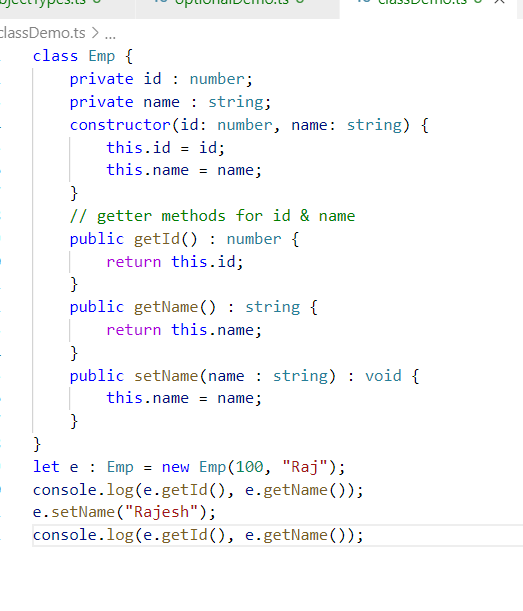
We have 3 access modifiers in typescript

* private: visible only within the class
* protected: visible with the class and can be inherited in the subclass or accessible only within the subclass
* public: visible everywhere, By default members are public

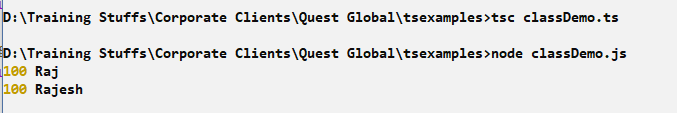
Note: Any variable or class or functions you create in typescript file will have local scope i.e., only within that script file you can access, if you want to access outside the script file you need to use script as modules

Creating class with setters & getters

classDemo.ts



Output:

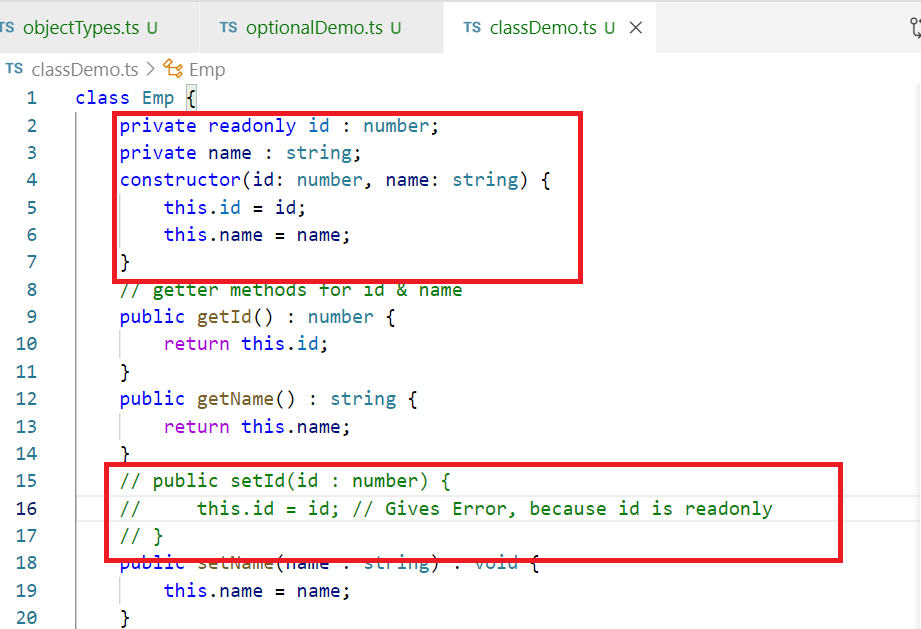


Class with readonly variables

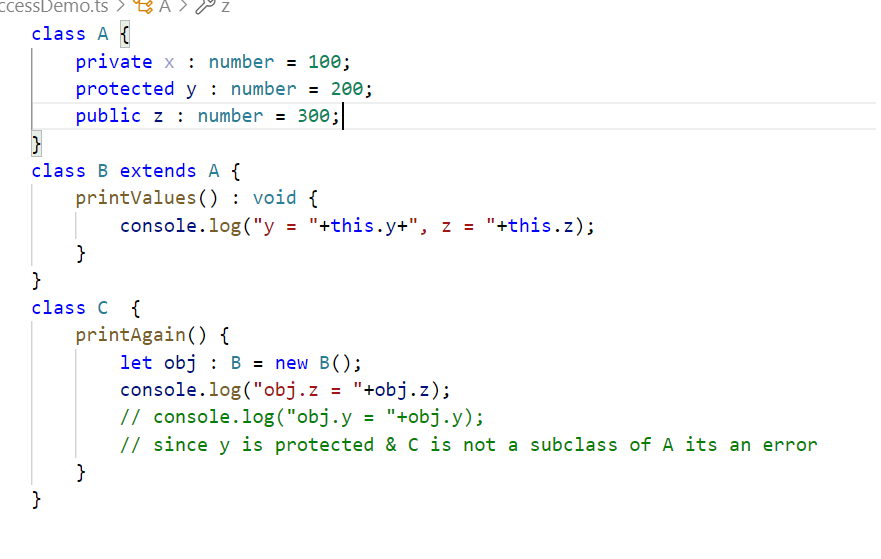
It is a variable which can’t modify once its initialized in the constructor

Note: const and readonly are not same, because const must initialized at the time of declaration only, whereas readonly variables can be initialized at the time or declaration or in the constructor, once the object is created, it can’t be modified

Note: creating setters for readonly gives error



Using access modifiers



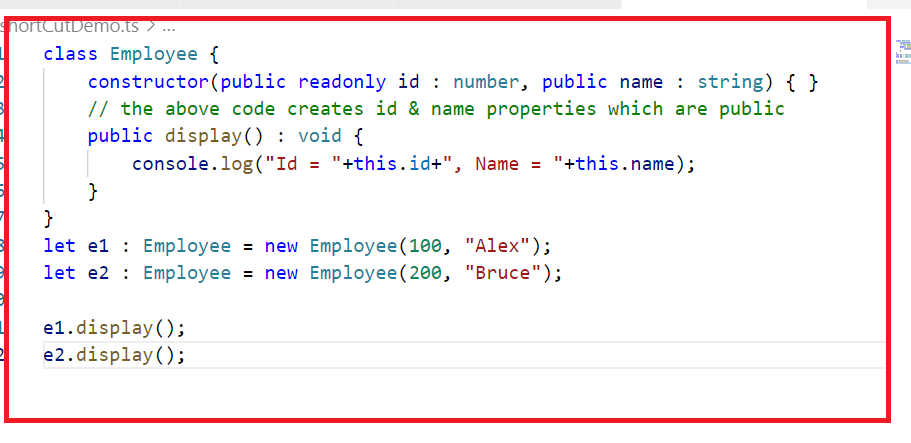
Short-cut initialization of the object

You can create a class without declaring properties & initializing the properties and make typescript to initialize by writing short-cut initialization constructor syntax, but the constructor parameters must have access modifiers so that you can access the property

class Employee {   
 constructor(public id : number, public name : string){}  
}

The above class will create id & name properties for Employee & also initializes its

shortCutDemo.ts



Output:

