Enum → Restrict a Value of Variable
→ To restrict user choice

```
002.basics > TS enum.ts > [@] hcSeat
1   enum SeatChoice {
2       AISLE = "aisle",
3       MIDDLE = 3,
4       WINDOW,
5       FOURTH
6   }
7
8   const hcSeat = SeatChoice.AISLE
```

Enum Sex {
M = "Male",
F = "Female",
T = 12; }

Now Sex can store only 3 options Available So we restrict user Choice by these !!

Default initial value for 1st value is 0 & then after Every Variable gets One More Value than previous one !!

Here in Seat Choice Middle has value 3 & so Window has Value 4 & Fourth has value of 5

Also Once you break pattern by Assigning some value Eg Enum Seat {
Lower = "lower",
Middle = "middle",

We Must tell which Value do just after ← { Upper Side Lower Side Upper
& rest after that (Side Lower & Side Upper) will get their Value by Upper one 3

If aisle = 1 Side Lower = 12 Side Middle = 13
If aisle = 1 " = 2 " = 3

Its compulsory to provide value of aisle & rest two sets value over its own

```
"use strict";
var SeatChoice;
(function (SeatChoice) {
    SeatChoice["AISLE"] = "aisle";
    SeatChoice[SeatChoice["MIDDLE"] = 3] = "MIDDLE";
    SeatChoice[SeatChoice["WINDOW"] = 4] = "WINDOW";
    SeatChoice[SeatChoice["FOURTH"] = 5] = "FOURTH";
})(SeatChoice || (SeatChoice = {}));
const hcSeat = SeatChoice.AISLE;
```

JS generated for this TS code
This is IIFE

```
1  const enum SeatChoice {
2      AISLE = "aisle",
3      MIDDLE = 3,
4      WINDOW,
5      FOURTH
6  }
7
8  const hcSeat = SeatChoice.AISLE
```

→

```
"use strict";
const hcSeat = "aisle" /* SeatChoice.AISLE */;
```

use this
Just put a const before Enum & JS code generated is very clear

Interfaces

Much more similar to type but here we have fn too

We can put variables readOnly. We know Java Interfaces like that there it just have fn declaration & variables

myInterface.ts > ...

```ts
interface User {
    readonly dbId: number
    email: string,
    userId: number,
    googleId?: string
}

const hitesh: User = { dbId: 22, email: "h@h.com", userId: 2211}
hitesh.email = "h@hc.com"
hitesh.dbId = 33
```

Simple Interface

→ Error as can't change it

Lets put fn inside Interface

Easily understandable

002.basics > TS interface.ts > ...

```ts
interface User {
    readonly dbId: number
    email: string,
    userId: number,
    googleId?: string
    // startTrail: () => string
    startTrail(): string
    getCoupon(couponname: string, value: number): number
}

interface User {
    githubToken: string
}

interface Admin extends User {
    role: "admin" | "ta" | "learner"
}

const hitesh: Admin = { dbId: 22, email: "h@h.com", userId: 2211,
role: "admin",
githubToken: "github",
startTrail: () => {
    return "trail started"
},
getCoupon: (name: "hitesh10", off: 10) => {
    return 10
}
}
hitesh.email = "h@hc.com"
```

Two styles of fn declaration

Another Method

Reopening of Interface → adding more properties

Type don't have fn in it !!

Inheritance

Reopening of Interface → adding more properties to it as per matching child class of it.

We can also Extend Multiple Interfaces

Differences Between Type Aliases and Interfaces

Type aliases and interfaces are very similar, and in many cases you can choose between them freely. Almost all features of an `interface` are available in `type`, the key distinction is that a type cannot be re-opened to add new properties vs an interface which is always extendable.

from documentation

Interface	Type
Extending an interface	Extending a type via intersections
Adding new fields to an existing interface	A type cannot be changed after being created

```ts
interface Animal {
  name: string;
}

interface Bear extends Animal {
  honey: boolean;
}

const bear = getBear();
bear.name;
bear.honey;
```

```ts
type Animal = {
  name: string;
}

type Bear = Animal & {
  honey: boolean;
}

const bear = getBear();
bear.name;
bear.honey;
```

```ts
interface Window {
  title: string;
}

interface Window {
  ts: TypeScriptAPI;
}

const src = 'const a = "Hello World"';
window.ts.transpileModule(src, {});
```

```ts
type Window = {
  title: string;
}

type Window = {
  ts: TypeScriptAPI;
}

 // Error: Duplicate identifier 'Window'.
```

Setting Typescript in Production

1st use Command tsc --init → It gives a config file

```
mohit@BOOK-N00DGRN68S MINGW64 ~/Desktop/programs/typescript/003.purets (main)
$ tsc --init

Created a new tsconfig.json with:

  target: es2016
  module: commonjs
  strict: true
  esModuleInterop: true
  skipLibCheck: true
  forceConsistentCasingInFileNames: true


You can learn more at https://aka.ms/tsconfig

mohit@BOOK-N00DGRN68S MINGW64 ~/Desktop/programs/typescript/003.purets (main)
```

Then we need to get npm init as we want node project so npm init -y

↳ so that it won't ask any Questions

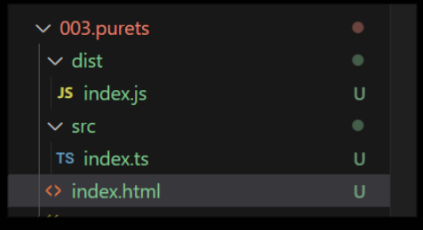
```
mohit@BOOK-N00DGRN68S MINGW64 ~/Desktop/programs/typescript/003.purets (main)
$ npm init -y
Wrote to C:\Users\mohit\Desktop\programs\typescript\003.purets\package.json:

{
  "name": "003.purets",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

get package.json

Now write index.html for that we create src folder In production you have 2 folder src & dist [distribution — This is served to End user]

We will have .ts file inside src folder & we want to produce .js file inside dist folder so for that we need to configure But lets 1st see folder structure

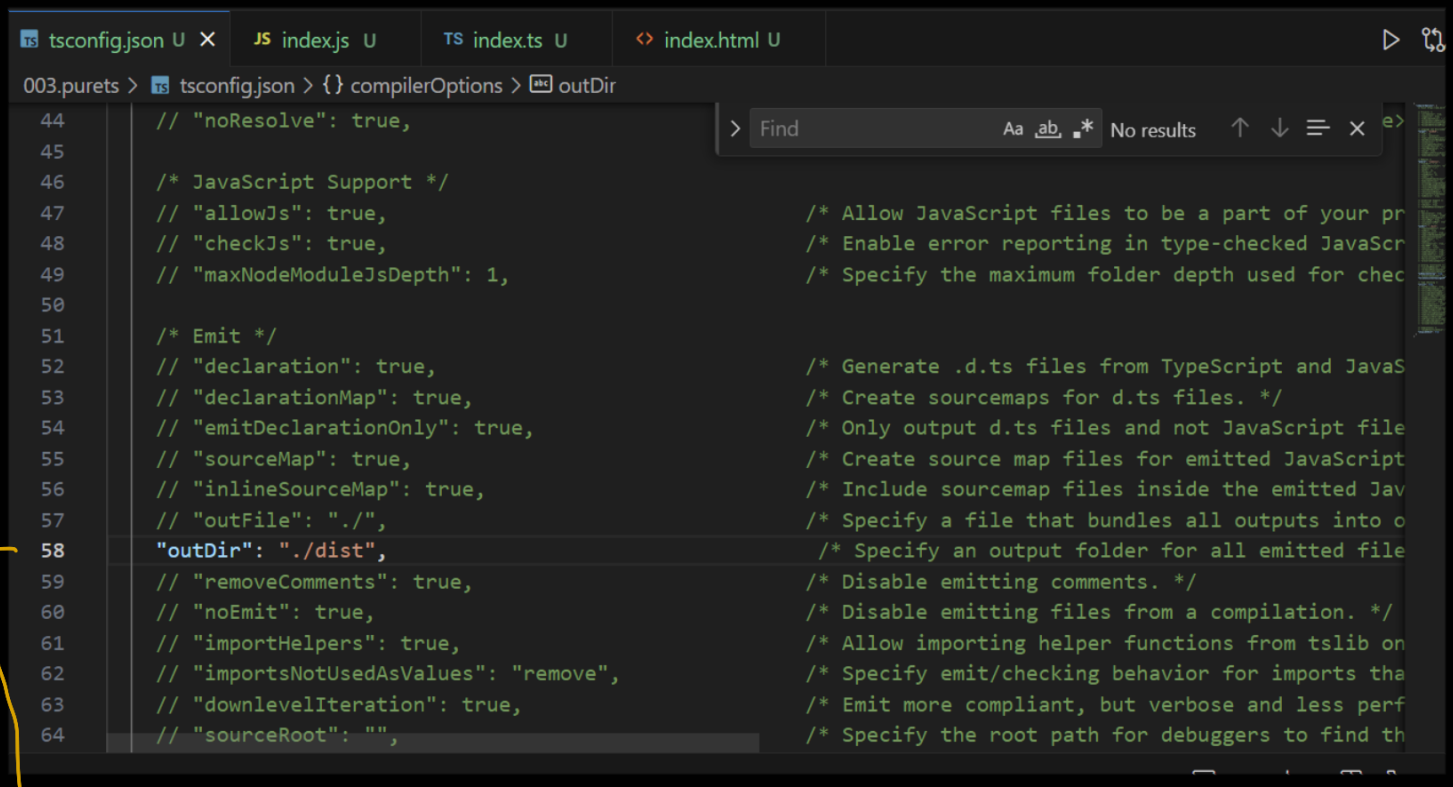


003.purets > index.html > html > body

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <script src="./dist/index.js"></script>
10 </body>
11 </html>
```

↳ we told to link js file

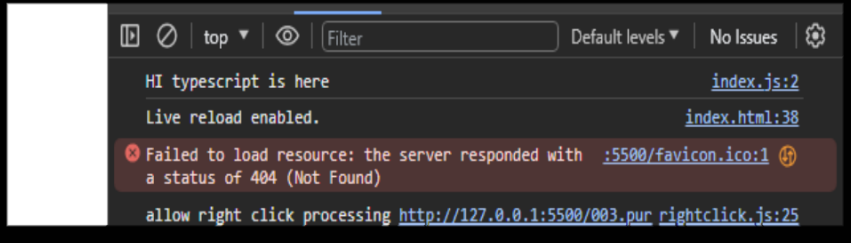
We go to ts config



→ Put there the destination of js file

Now after saving index.ts by tsc index.ts out Command tsc -w → Enables watch mode

} Generates o/p in dist



} See o/p of index.js

With tsc -w you make change to ts file will be reflected in ojs file & instantly in html file

Ctrl+C → to stop tsc -w

Classes

```
003.purets > src > TS index.ts > ...
1   class User {
2       public email: string
3       private name: string
4       readonly city: string = "Jaipur"
5       constructor(email: string, name: string){
6           this.email = email;
7           this.name = name
8
9       }
10  }
```

Simple Enough

object of class

```
49
50  const hitesh = new User("h@h.com","hitesh")
51  // hitesh.name
52
```

Although its typescript But still it is compiled as Javascript only so on console of browser you can change the city to 10 or true & there be no Error as its JS

Access Modifier → public & private

Read Only we get reassign them its like constant we dont want read only to be accessed outside class. so we can use private access Modifier private make sure variable can be used inside class only Default access Modifier is public

```
1  class User {
2      public email: string
3      name: string
4      private readonly city: string = "Jaipur"
5      constructor(email: string, name: string){
6          this.email = email;
7          this.name = name
8
9      }
10 }
11
12 const hitesh = new User("h@h.com","hitesh")
13 hitesh.city
```

In JS we put private by #

We can use # also in place of private

But private is giving us More detail

We can also define Variables like this in constructor ⟹

Just Syntax Changes

no need to do this. _ = _ in constructor here

```
class User {

    readonly city: string = "Jaipur"
    constructor(
        public email: string,
        public name: string
        ){
    }
}
```

Getters & Setters → Much like in JS

```
class User {

    protected _courseCount = 1

    readonly city: string = "Jaipur"
    constructor(
        public email: string,
        public name: string,
        // private userId: string
        ){
    }
    private deleteToken(){
        console.log("Token deleted");

    }

    get getAppleEmail(): string{
        return `apple${this.email}`
    }

    get courseCount(): number {
        return this._courseCount
    }

    set courseCount(courseNum) {
        if (courseNum <= 1) {
            throw new Error("Course count should be more than 1")
        }
        this._courseCount = courseNum
    }
}
```

Private Method

Method can be accessed by class only

while attended by Email

Setter don't have a return type as it don't return Anything

Protected Access Modifier & inheritance

Inheritance a subclass can get private property We want something that child class can access & rest Everything same as private

```
44  class SubUser extends User {
45      isFamily: boolean = true
46      changeCourseCount(){
47          this._courseCount = 4
48      }
49  }
50
```

So here we use protected See line 14 of user class i.e. protected _courseCount.

Interfaces

class implementing interface

```
003.purets > src > TS second.ts > ...
1   interface TakePhoto {
2       cameraMode: string
3       filter: string
4       burst: number
5   }
6
7   class Instagram implements TakePhoto {
8       constructor(
9           public cameraMode: string,
10          public filter: string,
11          public burst: number
12      ){}
13  }
14
```

class inheriting multiple interfaces !!

```
003.purets > src > TS second.ts > ...
1   interface TakePhoto {
2       cameraMode: string
3       filter: string
4       burst: number
5   }
6
7   interface Story {
8       createStory(): void
9   }
10
11  class Youtube implements TakePhoto, Story{
12      constructor(
13          public cameraMode: string,
14          public filter: string,
15          public burst: number,
16          public short: string
17      ){}
18
19      createStory(): void {
20          console.log("Story was created");
21
22      }
23  }
24
```

Abstract classes

Difference bhw Abstract class & interface?

Interface provide 100% abstraction But there can be implementation in Abstract class.

Abstract class mein abstract method jinki implementation nhi hoti bahi usko hi implementation krni pdhi hai, Also Abstract Class implement nhi Extend hota hai (More of Inheritance)

Abstract class ke object nhi bnte Sirf !! Ye Child class ki responsibility hai us class ke object bnana !!

003.purets > src > TS abstractclass.ts > TakePhoto

```ts
abstract class TakePhoto {
    constructor(
        public cameraMode: string,
        public filter: string
    ){}

    abstract getSepia(): void
    getReelTime(): number{
        //some complex calculation
        return 8
    }
}
class Instagram extends TakePhoto{
    constructor(
        public cameraMode: string,
        public filter: string,
        public burst: number
        ){
            super(cameraMode, filter)
        }

        getSepia(): void {
            console.log("Sepia");

        }
}

const hc = new Instagram("test", "Test", 3)

```

no implementation so declared as abstract

Implemented here

as this is child class So need to put super() and pass parameter to Super class

Abstract class = Interface + Simple class