Ejercicios

- Crear una carpeta en el disco duro.
- Por ejemplo: mkdir c:\ejercicios && cd c:\ejercicios && npm init -y
- Aparecerá el archivo package.json:

```
{
 "name": "ejercicios",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1",
   "run":"lite-server",
   "compilar": "tsc --watch",
   "limpiar": "del *.js"
 },
 "keywords": [],
 "author": "",
 "license": "ISC"
}
```

- En el cual hemos introducido tres tareas de npm
- teclear la orden tsc --init. Aparecerá el archivo tsconfig.json
- Crear la página index.html
- Crear los archivos main.ts y main.css al mismo nivel que index.html

Ejercicio0001

- Declarar algunas variables y constantes de los tipos de datos básicos y sacar sus valores por consola y/o index.html
- Comprobar las diferencias entre variables y constantes
- Experimentar creando algunos arrays de diferentes tipo, incluido any
- Definir algunos tuples
- Experimentar con tipos enumerados simple

Ejercicio0002

- Declarar un objeto con dos propiedades: nombre (string) y edad (number)
- Definir una función con la siguiente firma:
- function decodificar(p: {nombre:string, edad: number}) : {edad: number,nombre:string}
- Dentro de la función, obtener los valores del objeto que recibimos como parámetro mediante la desestructuración de objetos.
- Transformarlos de alguna forma y devolver un nuevo objeto.
- Mostrarlo por consola y/o la página index.html

Ejercicio0003

- Modificar el ejercicio 0002 empleando una interfaz para describir la estructura del objeto que la función decodificar recibe como parámetro.
- Transformarlos de alguna forma y devolver un nuevo objeto.
- Mostrarlo por consola y/o la página index.html

Ejercicio0004

- Definir la siguiente jerarquía: Animal, Perro, Gato usando interfaces
- Las dos últimas son subtipos de la primera
- Comparten la función: mostrar():string, y se les puede añadir propiedades y métodos específicos
- Experimentar con propiedades opcionales y de sólo lectura
- Crear un Animal[], introducir al menos un gato y un perro y sacar sus datos por consola y/o la página index.html

Ejercicio0005

- Implementar el ejemplo anterior empleando clases en lugar de interfaces
- Experimentar con clases y métodos abstractos
- Observar las diferencias si usamos clases en lugar de interfaces

Ejercicio0006

- Definir la noción de Conjunto
- Emplear funciones flecha cuando sea posible
- Usar genéricos si se considera necesario
- Un conjunto tiene las siguientes operaciones:
- nuevo → permite añadir un nuevo elemento al conjunto. Si el elemento ya existe devuelve false, de lo contrario lo almacena y retorna true
- buscar → busca un determinado objeto. Si lo encuentra lo devuelve, de lo contrario retorna null
- mostrar → imprime por consola todos los elementos del conjunto
- Siendo la constante c un conjunto de determinado tipo, observar lo que sucede con esta secuencia:
    - const m = {n:'abc', e:20};
    - c,nuevo(m);
    - c.nuevo(m);
    - c.mostrar();
- Y ahora, con esta otra:
    - c,nuevo(m);
    - c.nuevo({n:'abc', e:20});
    - c.mostrar();
- ¿Podríamos mejorar el código de alguna forma?

Ejercicio0007

- Definir una estructura con un nombre y una edad
- Definir un contenedor capaz de almacenar objetos con esa estructura, esto es, un repositorio. Representar el repositorio como una clase
- Crear los métodos siguientes:
- localizarPorNombre(n: string), devuelve un array
- cambiarTodasLasEdades(n: string), devuelve un array

- Resolverlo mediante clases y/o interfaces
- Emplear expresiones lambda (funciones flecha, en la medida de lo posible)

Ejercicio0008

- Rediseñar el ejercicio0007 para que la clase que representa el repositorio use genéricos
- Experimentar empleando namespaces