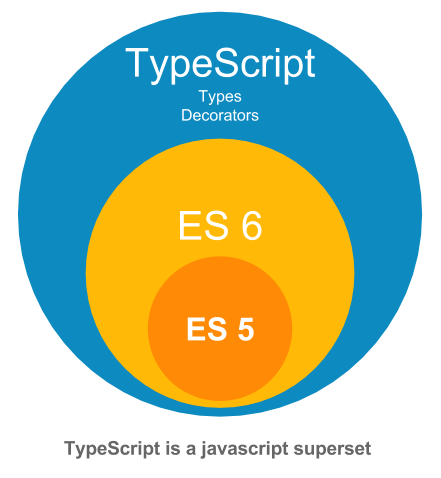
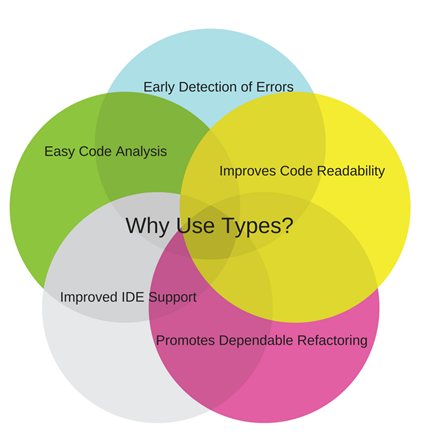
36: TypeScript

Introduction a TypeScript

Se conoce como un super set de Js



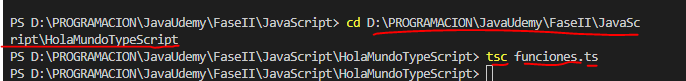


Permite la rrapida detección de errores, mejora de código …

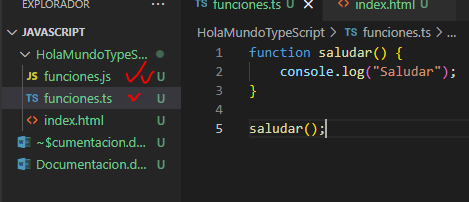
Instalación de TypeScript

Es el mismo código de Js pero para poder ejecutarlo primero se debe compilar ejemplo

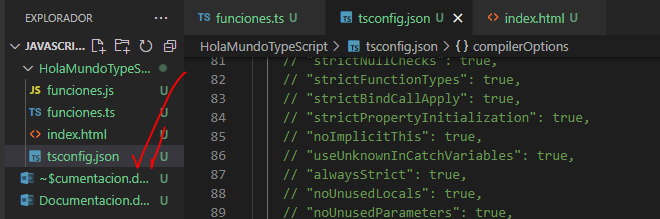
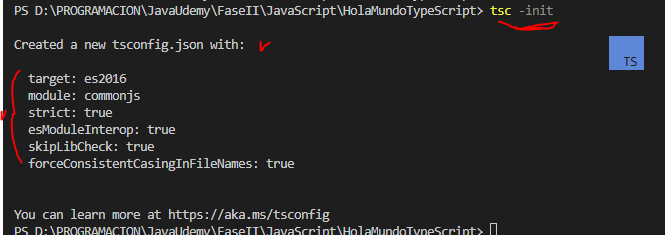
tsc funciones



Y automáticamente nos crea el archivo js



El siguiente paso es indicarle al ide que estamos trabajando con un proyecto ts



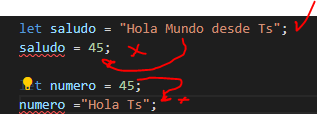
Siguiente paso converti los archivos en automatico de TS a JS

Se escribe en la consola tsc -w

En automático inicia la configuración.

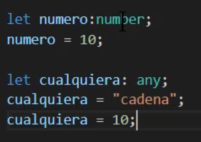
Definición de Variables con TypeScript

Las variables se declaran let y solo son remplazables por datos de del mismo tipo ejemplo



Si no le voy a dar un valor en especifico a la variable debo inicializarla con su tipo de dato.

Existe un tipo de dato que aplica para cualquier tipo de variable any

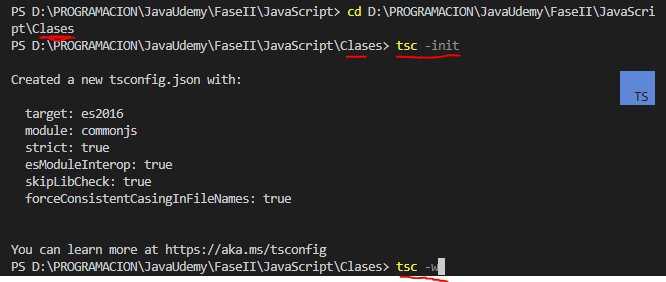


Otro tipo de variable es de tipo CONST que es un valor único y no se puede modificar. Es buena practica declarar la variable en MAYUSCULA para el const



 Definición de Clases y Objetos con TypeScript

Iniciamos el proyecto en una nueva carpeta



Un clase puede contener métodos estaitcos

Definición de Interfaces con TypeScript

Una interface es un contrato que debemos de respetar, conteniente ciertos atributos o métodos y para poder utilizarlos siempre se deben respetar la inicialización de atributos y métodos.

//EL SIMBOLO DE INTERROGACION SIGNIFICA EL EL ATRIBUTO ES OPCIONAL

EN COPNCLUSION PARA mi una interface es como un constructor que debe respectar los atributos de la interface para ser utilzada

Definición de Generics con TypeScript

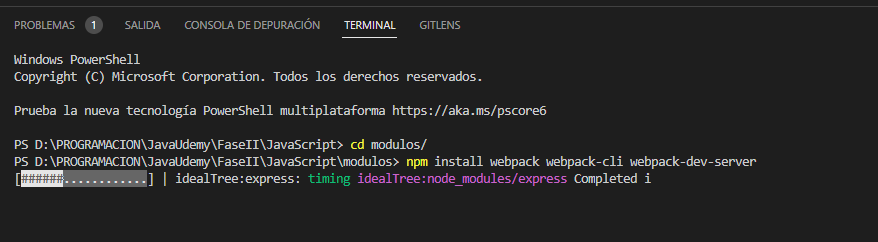
Utilizar un arreglo y especificar el tipo de dato que almacena dicho arreglo y utilizar el dicho arreglo

 Uso de Módulos con TypeScript

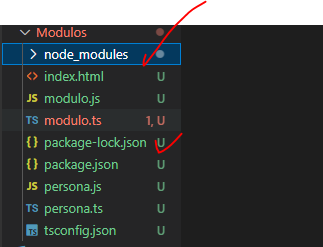
Útil si yo usar la calse de otro archivo entonces aplico la palabra reservada Export e import.

No solo eso es necsario necesitaremos de un SERVIDOR para que resuelva los modulos (BAJO EL CONCEPTO DE WEBPACK) y procedemos a intalarlo

npm install webpack webpack-cli webpack-dev-server

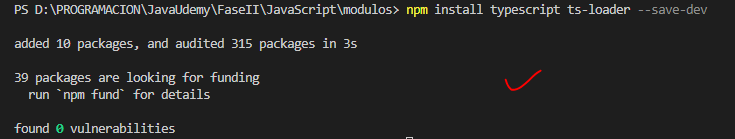


Se instala una carpeta y una archivo para que todo ya este OK

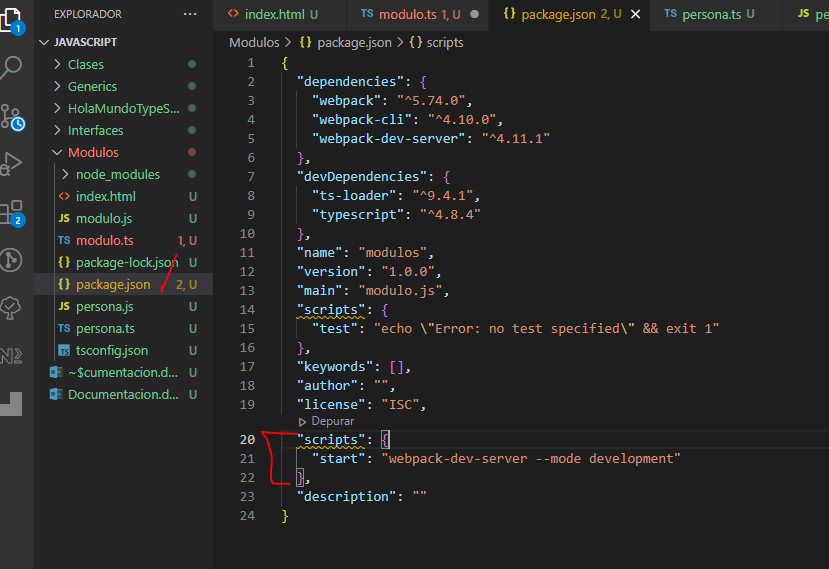


Ahora para cargar las clases instalamos el typescriptloger

npm install typescript ts-loader --save-dev



Ahora generamos el archivo package.json con el siguiente comando npm init -y para luego modificarlo y decirle al servidor como vamos a trabajar, agregar las líneas script



  "scripts": {

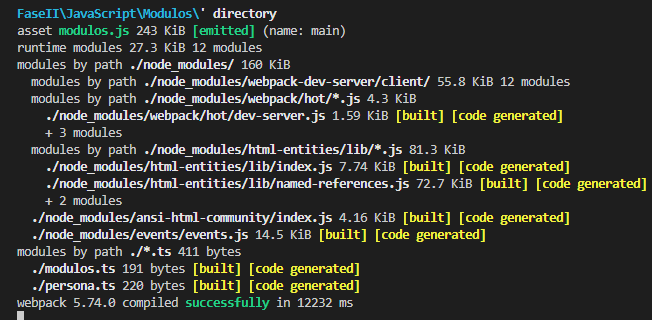
    "start": "webpack serve --mode development"

  }

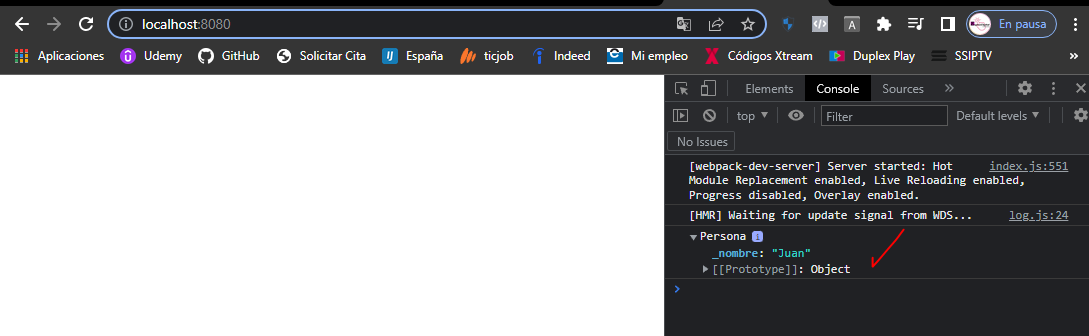
Creamos el arcvhivo webpack.config.js este archivo contendrá el código par habilitar el archivo principal y realice el funcionamiento del servidor

1. const path = require('path');
3. module.exports = {
4. entry: path.join(\_\_dirname, '/modulos.ts'),
5. output: {
6. filename: 'modulos.js',
7. path: \_\_dirname
8. },
9. module: {
10. rules: [
11. {
12. test: /\.tsx?$/,
13. loader: 'ts-loader',
14. exclude: /node\_modules/,
15. },
16. ]
17. },
18. resolve: {
19. extensions: [".tsx", ".ts", ".js"]
20. },
21. devServer: {
22. static: {
23. directory: path.join(\_\_dirname, '/')
24. }
25. },
26. };

Luego procedemos a iniciar el servidor npm start



Todo OK



 Funciones de Flecha con TypeScript

Clase op y conceptual

Decoradores con TypeScript

El uso de decoradores se puede aplicar a clase a métodos y también a atributos o propiedades de nuestra clase. Como son decoradores y están en pruebas para probarlos debemos iniciar por la lista de comando el tsc -init y el tsc -w

En la prueba se ve como manda error ya que es una característica experimental, pero igualmente deja operar

