14.1 - Typescript 1

TODO:

1. Take input in TS
2. Write a function to calculate sum
3. Write a function which takes another function as an input and runs it after 1 second

- You should always use Typescript code.
- Compiled languages mean your code gets converted to a binary and that binary is run.
    - If there is an error, the code will not be compiled into a binary but will rather show the errors.
        - These errors are called compile-time errors.
- On the other hand, interpreted (not compiled) languages files are compiled as they are run.
    - This leads to runtime errors, there is no safety net of the compiler identifying issues.
- Any javascript code is valid typescript code since typescript is a superset of javascript. But any typescript code is not valid javascript.
- Typescript code is eventually compiled to Javascript and that is what is run on the browsers.
    - As typescript is compiled/converted to javascript, if there is an error, the conversion will fail and there will be no javascript file for the corresponding typescript file.
- Typescript is used everywhere over javascript.
- There are many compilers to compile Typescript like tsc, esbuild etc.
- Npx = npm executable
- Ideally, run npm install --dev typescript as it should be ideally a dev dependency and you don't need it during runtime.
    - npx tsc --init
    - let x: number = 1
- To run TS code:
    - npx tsc -b (this outputs the javascript file of your typescript file index.ts)
    - node index.js or whatever file name.
- You can give hybrid types:
    - let x: number | string = 1
    - then x = "hello" is valid
- There is a special type in typescript: any
    - This includes ALL types.
    - If you want a variable to be any, you have to make it explicitly any otherwise typescript will complain.
    - **AVOID** using ANYs, otherwise it defeats the purpose of using typescript.
- Return type for functions in TS: function sum(a: number, b: number): number {}
    - This way, if you assign the output of a function to a new variable, you don't have to define that variable's type - it gets automatically set by inference.
- 

    - Line 1 is how you define a function type in the parameter (takes no output and returns void).

- The organization that handles javascript's updates is ECMA, and they have the ECMAscript standard (extra information).
    - Based on these standards, all web browsers update themselves to fit the ECMA standards.
- TS Config file options:
    - Target - which year ECMA version should the code be compiled to.
    - Whenever you run 'npx tsc -b' to build your TS files, it creates counter-JS files for ALL TS files.
    - You should not push the compiled down javascript files to github because your co-developers dont need to see that code, they just need the ts code which they can themselves compile to run.
        - Put all your TS files in /src and create another folder /dist or /build for your js files.
        - Add "rootDir": "./src" and "outDir": "./dist" to your ts config file.
        - Lastly, add the /dist folder to your .gitignore.
- Object type as argument in a function:
    -
```
function greet(user: {
    name: string,
    age: number
}) {
    console.log("hello " + user.firstName)
}
```
- The 'type' for objects is called interfaces, where:
    - const user = {name: "abhi"} becomes
        - interface User = {name: "abhi"}
- The ONLY caveat of typescript is:
    - Append a type to each variable, argument, and function with ": type" (int in TS is: number).
    - Functional arguments are written as: function abcd(fn: (a: string) => string)): number {}
        - Which means that abcd takes a function fn as an argument (which itself has a which is a string as an argument) and which should return string, while function abcd returns a number.
- Interface is a custom type:
    - interface UserType {name: string, age: number}
- Types are almost exactly the same as interfaces, they just give you a few extra abilities:
    - Union types: user: String | UserType
    - type StringOrNumber = String | Number
    - Intersection: type lead: Employee & Manager
        - Manager and Employee can be interfaces but lead has to be a type.
-