14.2 - Typescript 2

TODO:

1. **Make a field in TS optional**
2. **Create two types user and admin, create function that takes either user or admin as input, and returns a string “welcome [name]”**
3. **Given an array of users, filter out the legal aged users.**

- To make a field optional:
    - Add question mark: user {address?: string}
        - Address is an optional field.
        - Note: either the complete address is optional, or you need to make every inner parameter optional if you want to allow partial address.
- An interface can be implemented as a class.
- In functions, you can write as func: () => string OR func: (): string <- BOTH mean the same.
- Interface does not exist in javascript, it only exists in typescript.
- Though you can create objects from interfaces directly as we have done before, classes give you extra advantages, like inheritance, polymorphism etc.
    - Though usually a class extends an abstract, not an interface, but it is possible for a class to implement an interface too.
- While implementing an interface, your class can add extra variables or functions, but you need a minimum of what the interface has.
- Classes ‘extend’ each other <- meaning that the class which extends the first class automatically gets access to the original class’s functions and variables (without needing to redefine them).

```
class Shape {
  area() {
    console.log("hi i am area");
  }
}

class Rectangle extends Shape {
  width: number;
  height: number;

  constructor() {
    super()
    this.width = 1;
    this.height = 2;
  }
}
```

    - Shape becomes the ‘parent’ of the Rectangle.
- On the other hand, ‘implements’ means that the class follows the blueprint to IMPLEMENT the blueprint.
- A class can implement AND extend at the same time.
- Note: you don’t have to write the function keyword in a class when defining a class: isLegal() {return true}.
- Abstract class (also a blueprint):

-
```
abstract class User {
  name: string;
  constructor(name: string) {
    this.name = name;
  }

  abstract greet: () => string;
}
```
- The implementation of an abstract class and an interface by another class is exactly the same. The only difference is, in abstract classes, you can also specify default implementations for the functions in case the implementing class does not implement it.
- On the other hand, a type lets you define a custom type but also lets you do other things like union, AND operator etc.
    - And, you cannot implement a type but you can implement an interface.
- Array in TS: int[] nums (in java) = **nums: number []** (in TS).
- Differences between type and interface:
    - Interfaces can be implemented by classes, types cannot be.
    - Types can use unions and intersections, interfaces cannot.
- When using express with TS, you have to install @types/express (so install express' types along with it).
- Tip: Assigning the same variable name even in two different files (at the same directory level) will give you an error, so use different names.
- An interface can extend another interfaces, or even multiple interfaces using: ***interface A extends B, C{}*** (which basically is the same as a class extending another class - ie, gaining the properties of the parent interface).