Typescript features to consider

- Intersection Type is just two interfaces combined with the "&" sign to mention that the result should have a type of both the combined interfaces.
- keyof operator takes an object and produce a string or numeric literal union of its keys. eg. type P = keyof object1;
- type assertion: is when we know about the type but TypeScript don't, we usually use "as 'type'" to assert a type.
- typeof type guard: when we have multiple option of types and we wanna make sure that what we get is a type of single type, we can use the typeof guard before trying to do anything to it. eg. if(type of prop === "string"){prop.charAt}.....
- extends: allows us to extend an interface by adding additional types.

```
interface BasicAddress {
name?: string;
street: string;
city: string;
country: string;
postalCode: string;
}
interface AddressWithUnit extends BasicAddress {
unit: string;
}
```

- if we don't wanna explicitly specify the type for Reuse-ability purpose ae can use Generics which lets us create type variables.

```
function createPair<S, T>(v1: S, v2: T): [S, T] {
return [v1, v2];
}
console.log(createPair<string, number>('hello', 42)); // ['hello', 42]

// on type aliese

type Wrapped<T> = { value: T };

const wrappedValue: Wrapped<number> = { value: 10 };
```

- we can provide default value incase there won't be. eg. <T = number>
- unknown type which alternative to any type but safer when we don't know right know.
- never type to throw error when assigned the never type.
- readonly string[] will make it to be only readable.
- there are utility types that could be very useful like: (partial, required, omit, read-only, exclude, pick....)