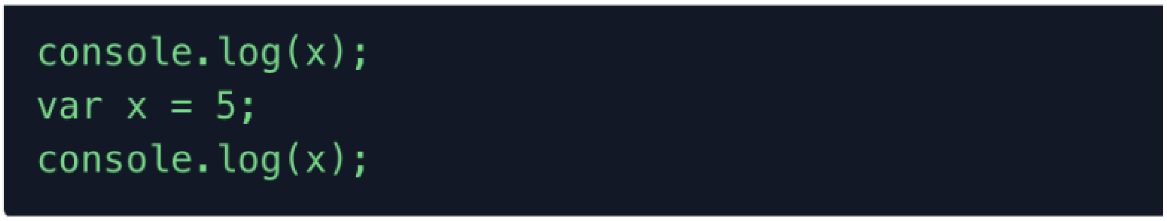
WEEK 1 SOLUTIONS:

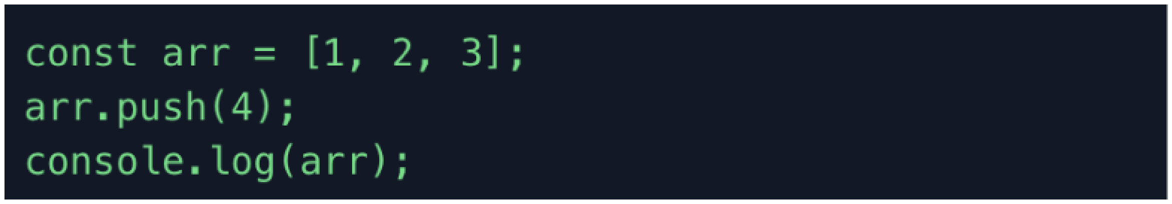
1. What will be the output of this code?



```
console.log(x);
var x = 5;
console.log(x);
```

OUTPUT:
undefined
5

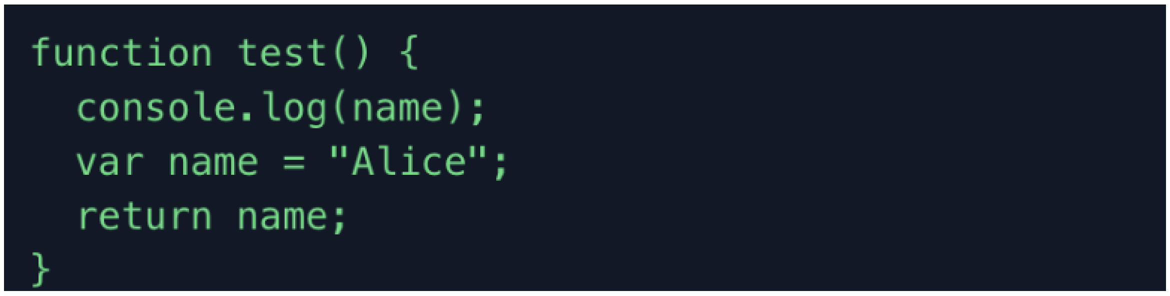
2. What happens when you run this code?



```
const arr = [1, 2, 3];
arr.push(4);
console.log(arr);
```

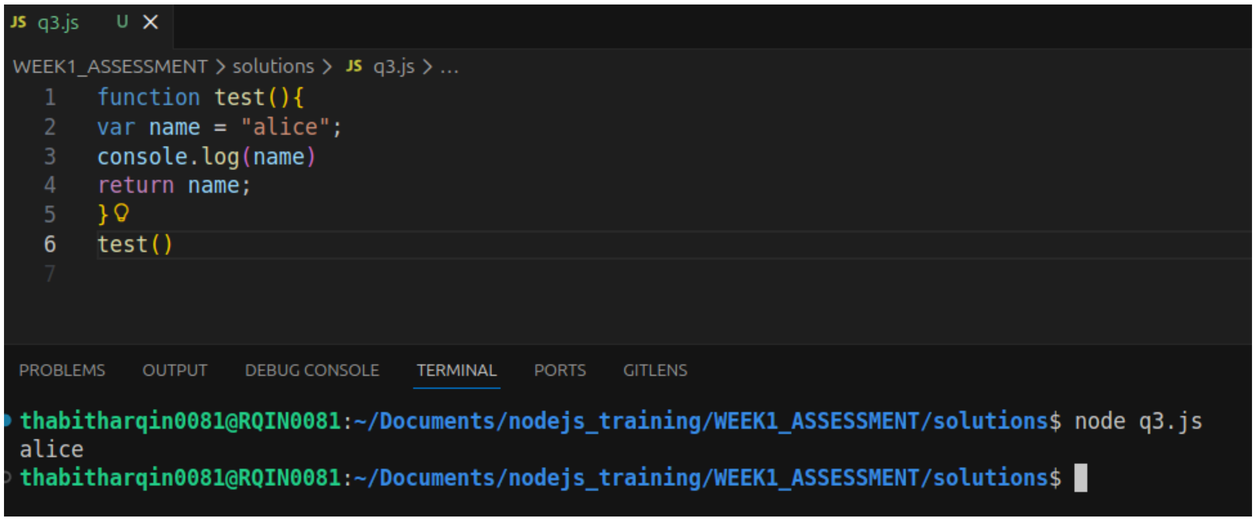
OUTPUT:
[1,2,3,4]

3. Fix this code to avoid hoisting issues:



```
function test() {
  console.log(name);
  var name = "Alice";
  return name;
}
```

OUTPUT:



q3.js U ×

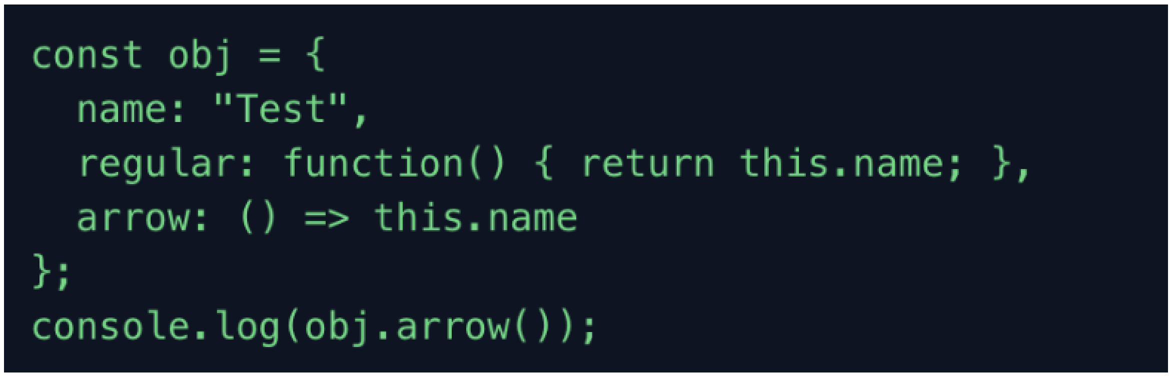
WEEK1_ASSESSMENT > solutions > q3.js > ...

```
function test(){
var name = "alice";
console.log(name)
return name;
}
test()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$ node q3.js
alice
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$
```

4. What is 'this' in the arrow function?



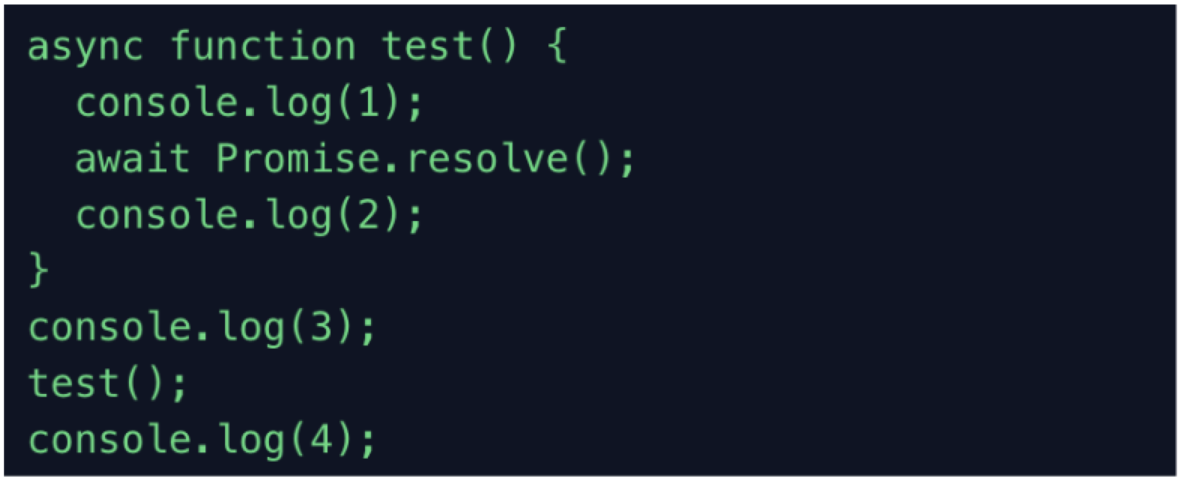
```
const obj = {
  name: "Test",
  regular: function() { return this.name; },
  arrow: () => this.name
};
console.log(obj.arrow());
```

OUTPUT:
“This” refers to nothing in the Arrow function so it prints undefined. But normally this refers to the object .

5. Write an arrow function that takes two numbers and returns their sum:

OUTPUT:
Const sum = (a,b)=>{return a+b}
console.log(sum(2,3))

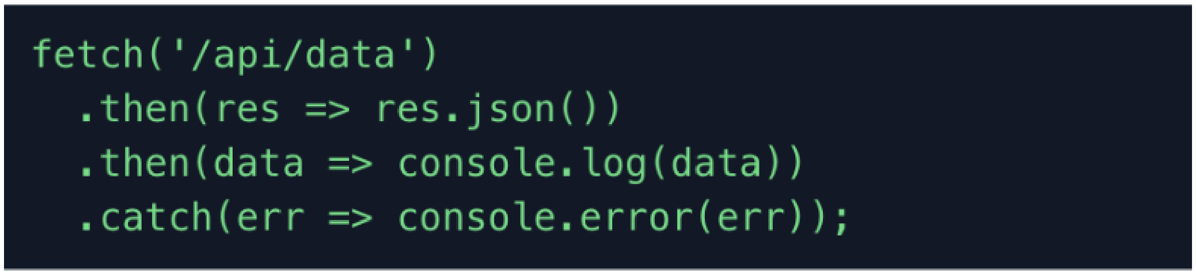
6. What will this output?



```
async function test() {
  console.log(1);
  await Promise.resolve();
  console.log(2);
}
console.log(3);
test();
console.log(4);
```

OUTPUT:
3
1
4
2

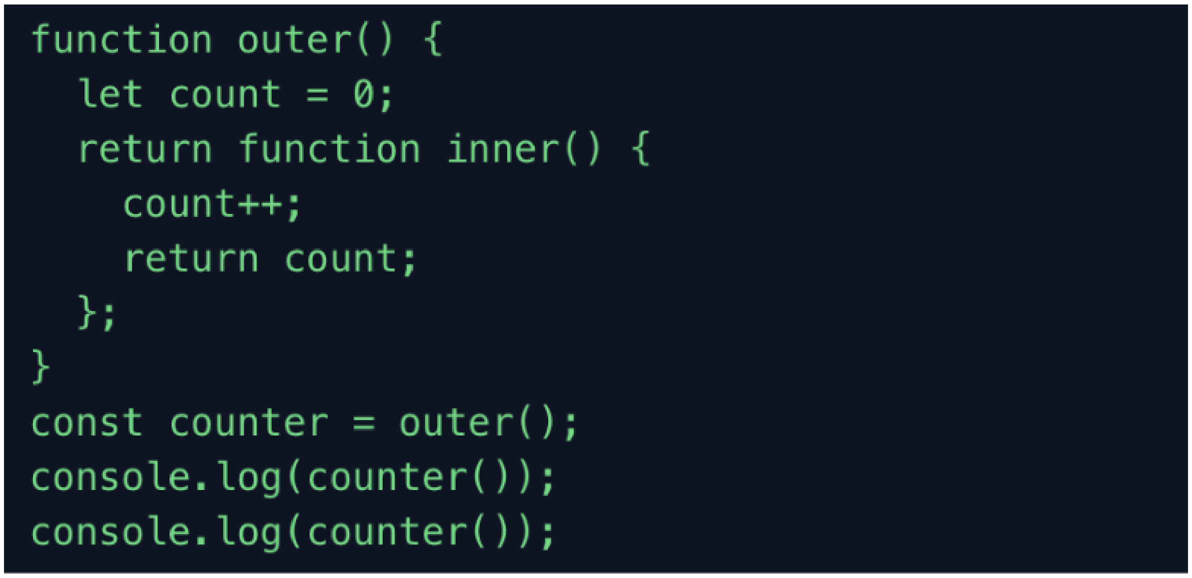
7. Convert this Promise chain to async/await with try/catch:



```
fetch('/api/data')
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error(err));
```

OUTPUT:
async function apiData(){
try{
const fetched = await fetch('/api/data')
const data = await fetched.json()
console.log(data)
}catch(err){
console.error(err)
}}
apiData()

8. What will be logged?



```
function outer() {
  let count = 0;
  return function inner() {
    count++;
    return count;
  };
}
const counter = outer();
console.log(counter());
console.log(counter());
```

OUTPUT:
1
2

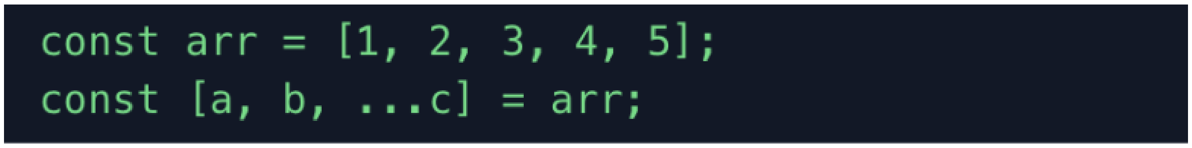
9. Create a function that returns a private counter (increment, decrement, getValue methods):

OUTPUT:
Function outer(){
Let count = 0;
Return{
increment(){
Count++
Return count;
},
decrement(){
Count–
Return count;
},
getValue(){
Return count;
}
}
}
Const counter = outer();
console.log(counter.increment())
console.log(counter.decrement())

console.log(counter.increment())

It prints 1, 0, 1

10. What is the value of c?



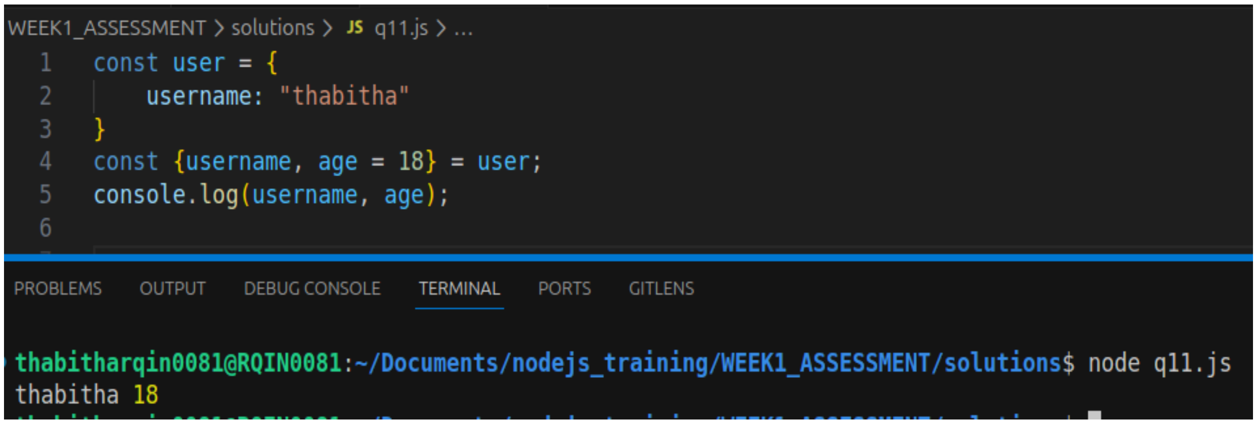
```
const arr = [1, 2, 3, 4, 5];
const [a, b, ...c] = arr;
```

OUTPUT:
[3, 4, 5]

11. Destructure this object to get name and age with default age of 18:

```
const user = { name: "Bob" };
```

OUTPUT:

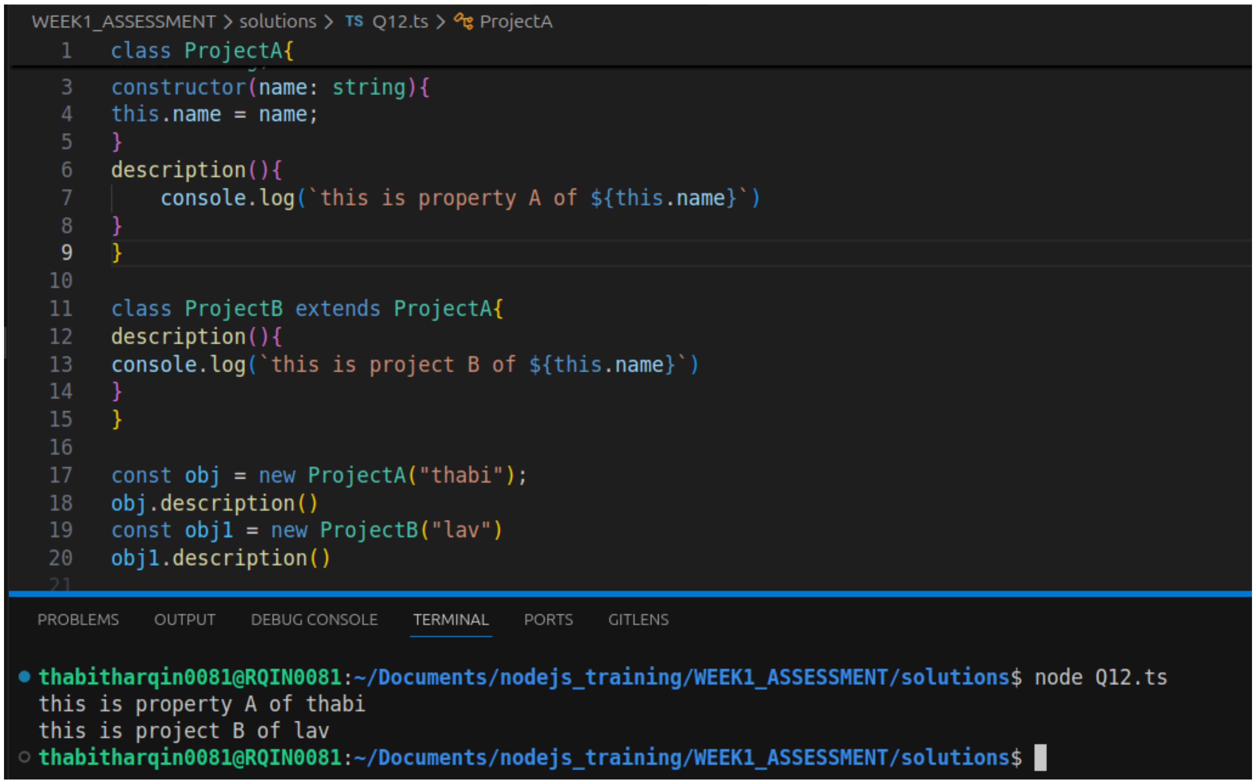


```
const user = {
    username: "thabitha"
}
const {username, age = 18} = user;
console.log(username, age);
```

```
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$ node q11.js
thabitha 18
```

12. Create a class 'ProjectA' with name property and description() method, then extend it with 'ProjectB' class that overrides description():

OUTPUT:



```
class ProjectA{

constructor(name: string){
this.name = name;
}
description(){
    console.log(`this is property A of ${this.name}`)
}
}

class ProjectB extends ProjectA{
description(){
console.log(`this is project B of ${this.name}`)
}
}

const obj = new ProjectA("thabi");
obj.description()
const obj1 = new ProjectB("lav")
obj1.description()
```

```
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$ node Q12.ts
this is property A of thabi
this is project B of lav
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$
```

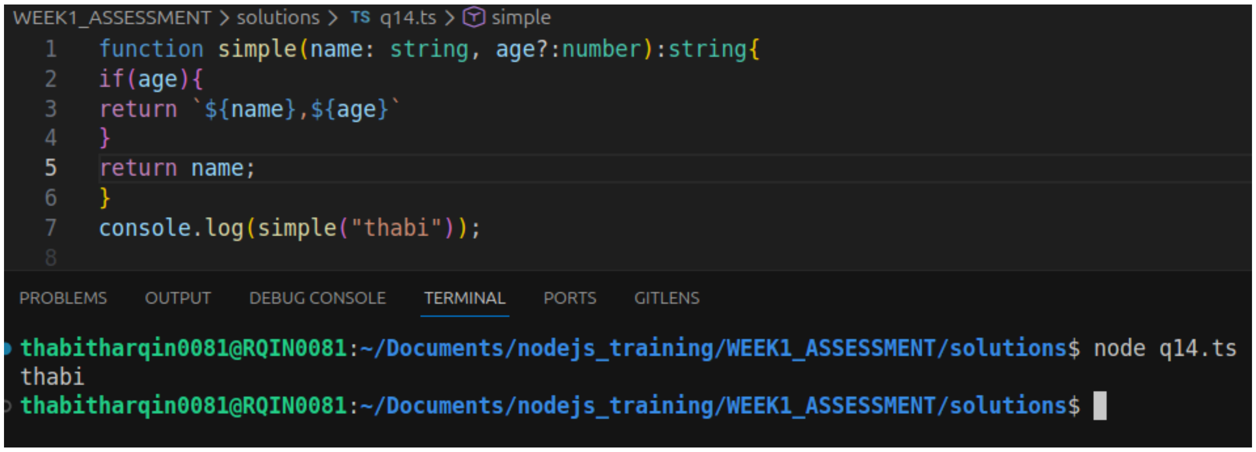
13. Which is the correct type annotation?

```
const numbers: ??? = [1, 2, 3];
```

OUTPUT:
Const numbers: number[] = [1,2,3];

14. Create a function that takes name (string) and age (optional number) and returns a string:

OUTPUT:



```
function simple(name: string, age?:number):string{
if(age){
return `${name},${age}`
}
return name;
}
console.log(simple("thabi"));
```

```
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$ node q14.ts
thabi
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$
```

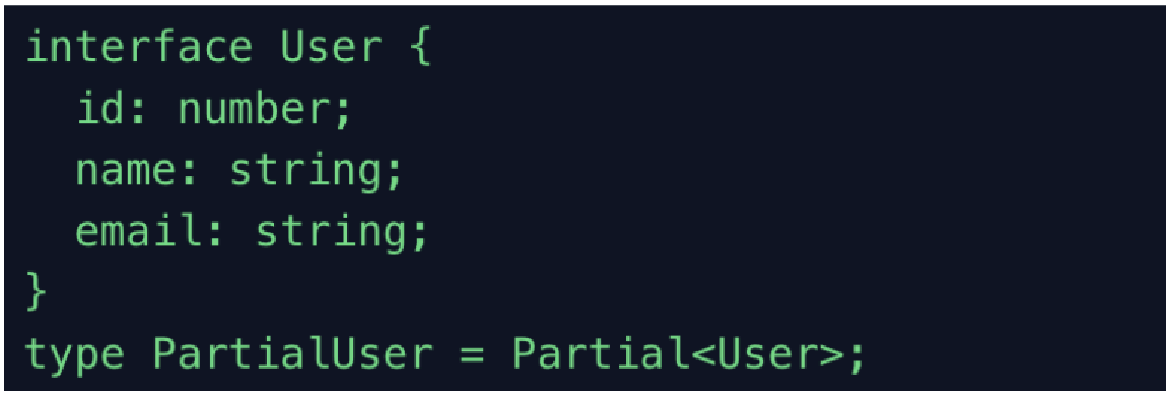
15. Define an interface 'User' with id (number), name (string), and optional email (string):

OUTPUT:
Interface User{
Id: number;
Name: string;
Email?: string;
}

16. What's the difference between interface and type alias?

OUTPUT:
Interface is applied to object but type alias can be applied to object, union, tuple.
In interface property can later be added but type does not allow that.
Interface uses extends, type uses & operator to take another type

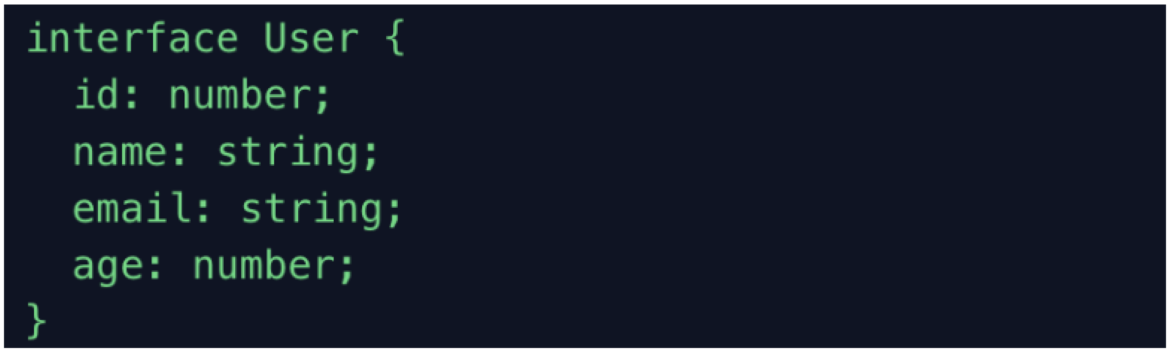
17. What does Partial<User> do?



```
interface User {
  id: number;
  name: string;
  email: string;
}
type PartialUser = Partial<User>;
```

OUTPUT:
It is not mandatory to have all property, partially it takes whatever property it comes.

18. Use Pick utility to create a type with only 'name' and 'email' from User interface:



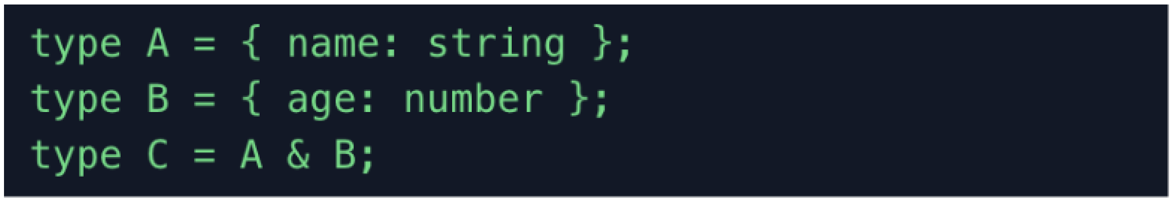
```
interface User {
  id: number;
  name: string;
  email: string;
  age: number;
}
```

OUTPUT:
Const user: User = Pick<User, “name”|”email”>
This user accepts only these variable

19. Create a function that accepts string | number and returns its length (for string) or value (for number):

OUTPUT:
Function simple(num: string | number):number{
if(typeof num === “string”)
Return num.length
}
Return number;

20. What is the result of this intersection type?

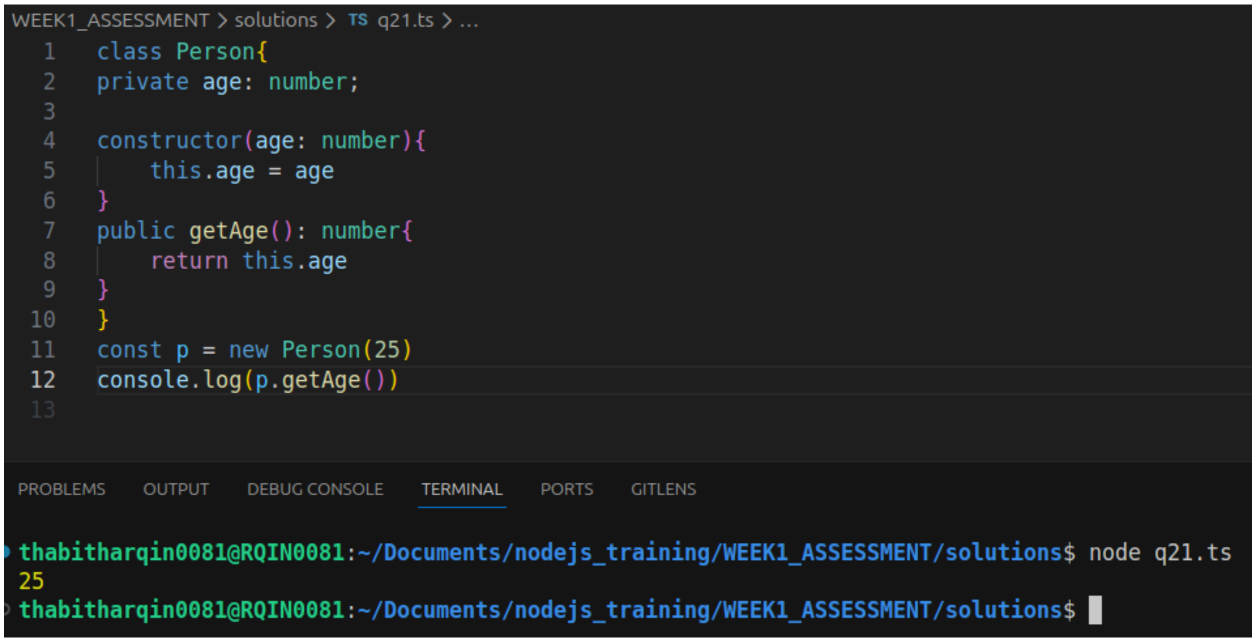


```
type A = { name: string };
type B = { age: number };
type C = A & B;
```

OUTPUT:
Now type C has both the property name and age of A and B and both are required.

21. Create a class 'Person' with private property 'age' and public method 'getAge()':

OUTPUT:



```typescript
class Person{
private age: number;

constructor(age: number){
    this.age = age
}
public getAge(): number{
    return this.age
}
}
const p = new Person(25)
console.log(p.getAge())
```

```
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$ node q21.ts
25
thabitharqin0081@RQIN0081:~/Documents/nodejs_training/WEEK1_ASSESSMENT/solutions$
```