COMP101: Review Session 3

With your boy, Mason

Announcements

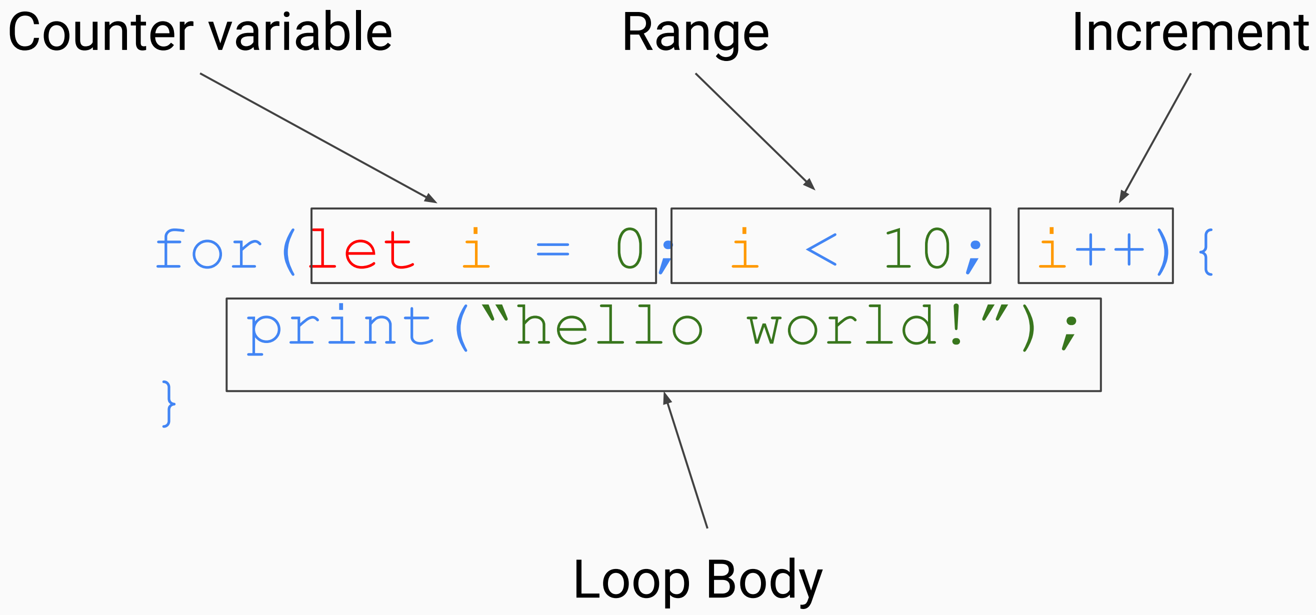
- Tony taught computer science fundamentals so much this week that he got sick and couldn’t be with us! Send him love!
- Quiz tomorrow: March 7, 2019. The focus will be on functions, looping, arrays, and scoping rules. Then: don’t think about computers for a week!
- Literally that’s it!

For-loops: like while loops, but better

- Task: Print “hello world!” ten times.

```
for(let i = 0; i < 10; i++){
   print(“hello world!”);
}
```

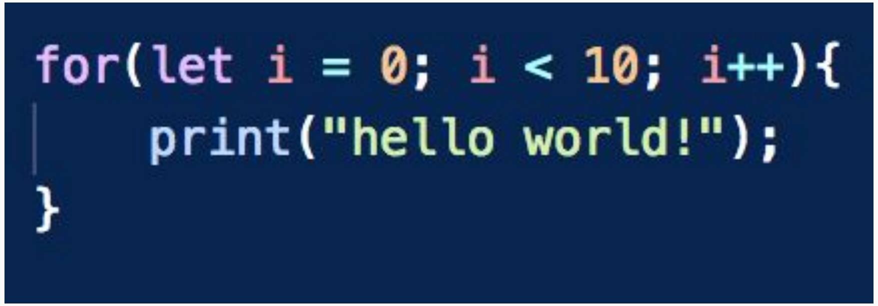
for-loops



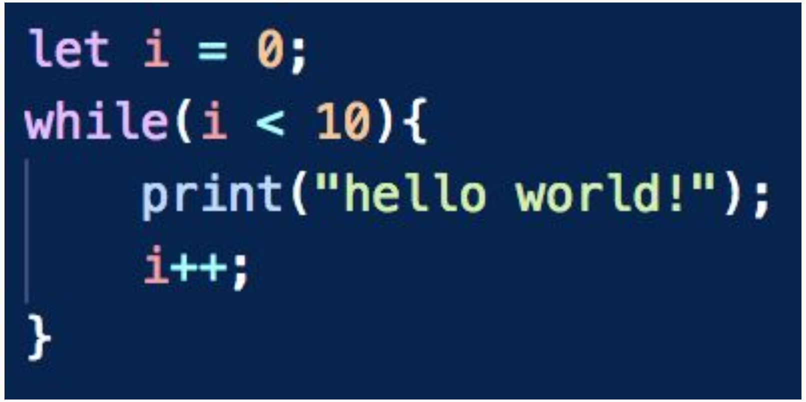
```
for(let i = 0; i < 10; i++){
   print(“hello world!”);
}
```

for-loops

These loops do the exact same thing!



```
for(let i = 0; i < 10; i++){
    print("hello world!");
}
```



```
let i = 0;
while(i < 10){
    print("hello world!");
    i++;
}
```

What are functions?

- Broadly speaking, a function has some input and some output.
- The types of the inputs and the output must be specified in the function definition.
- The function definition contains logic which processes the provided inputs and returns an output.

Adding Two Numbers

- It’s not that difficult to add two numbers in TypeScript, but what if we wanted to add *any* two numbers instead of hardcoding specific ones? We can write a function!

```
let i = 5 + 6;
let j = 3 + 9;
```

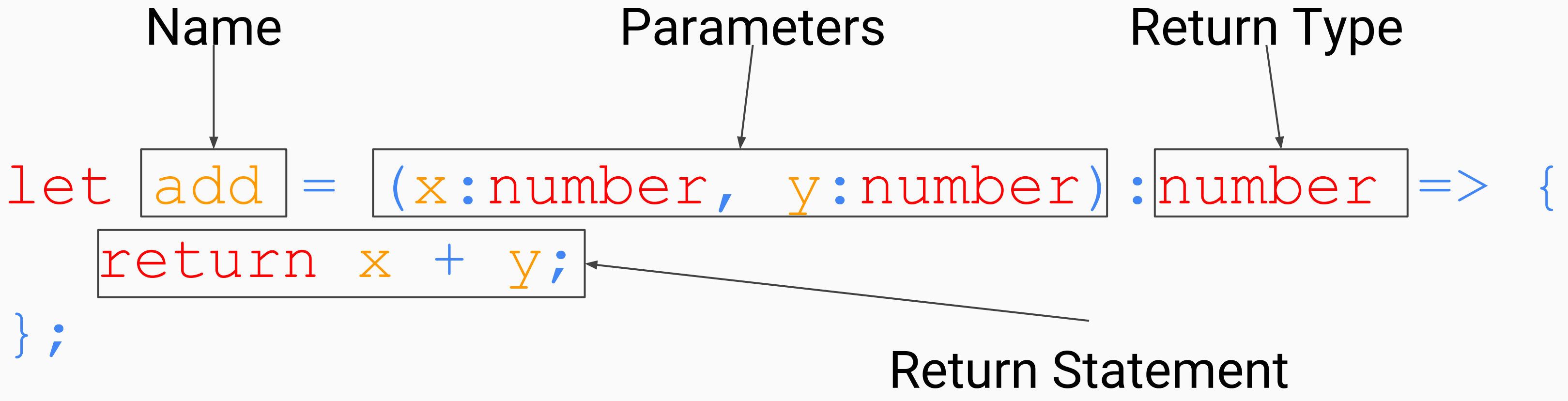
The anatomy of a Function

- Task: Create a procedure that can add two numbers

```
let add = (x:number, y:number):number => {
    return x + y;
};
```

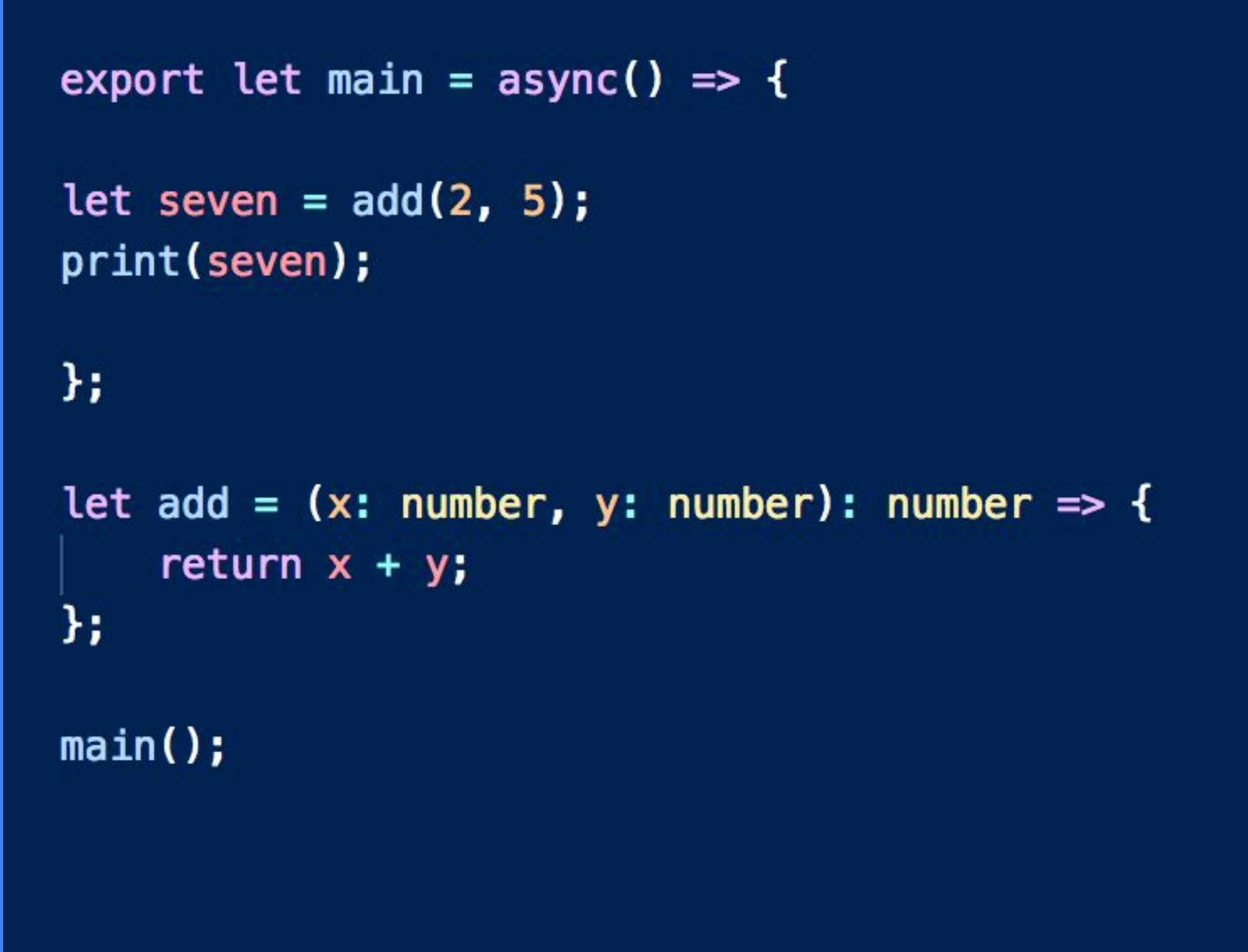
The anatomy of a Function

- Task: Name the different parts of the “add” function.



```
let add = (x:number, y:number):number => {
    return x + y;
};
```

The “add” Function In Action



```
export let main = async() => {

let seven = add(2, 5);
print(seven);

};

let add = (x: number, y: number): number => {
    return x + y;
};

main();
```

Jump to “main”

Call “add”

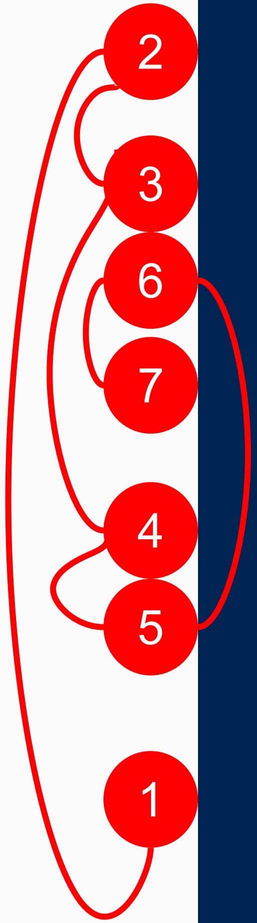
Print “seven”

Stop executing

Jump to “add”

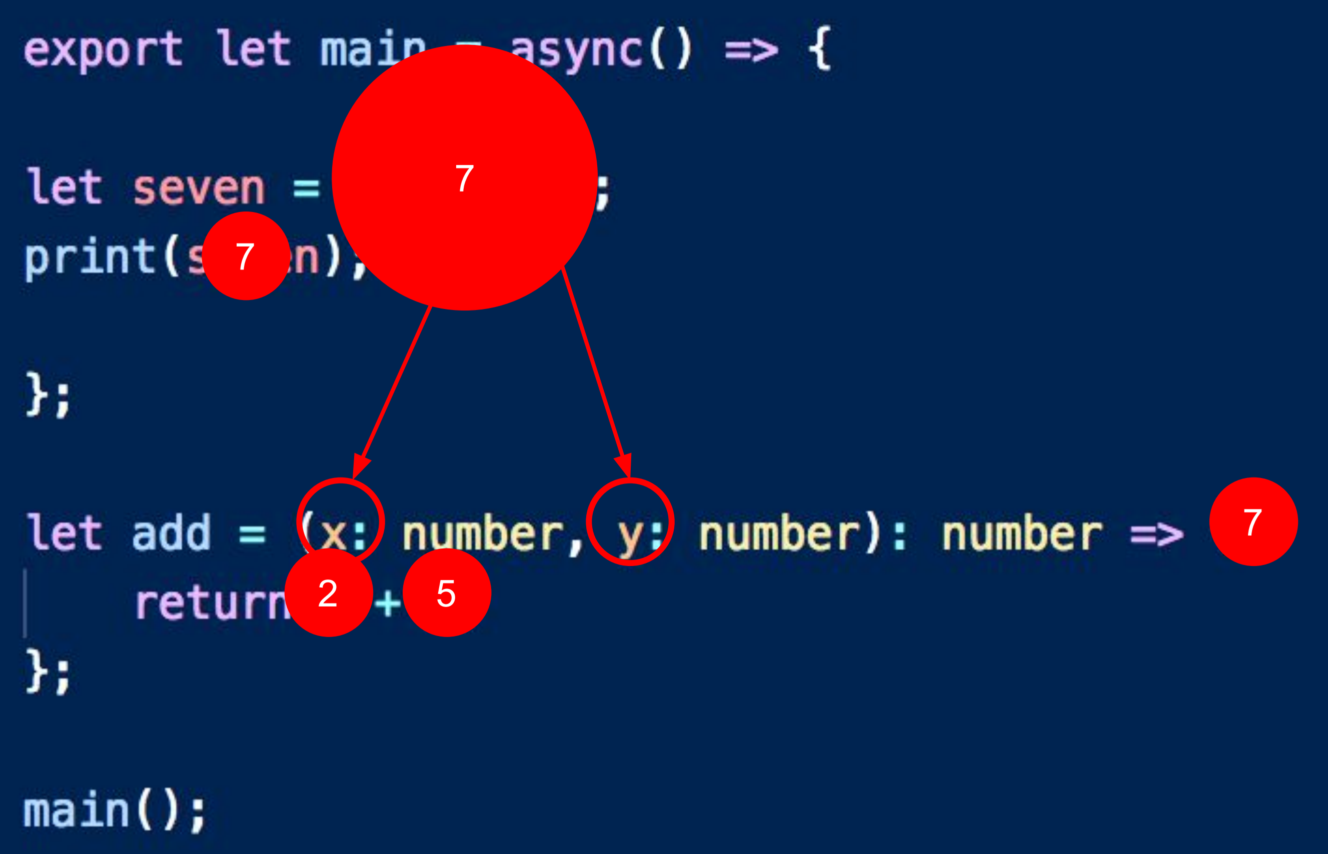
Execute “add”

Call “main”



```
export let main = async() => {

let seven = add(2, 5);
print(seven);


};


let add = (x: number, y: number): number => {
    return x + y;
};

main();
```

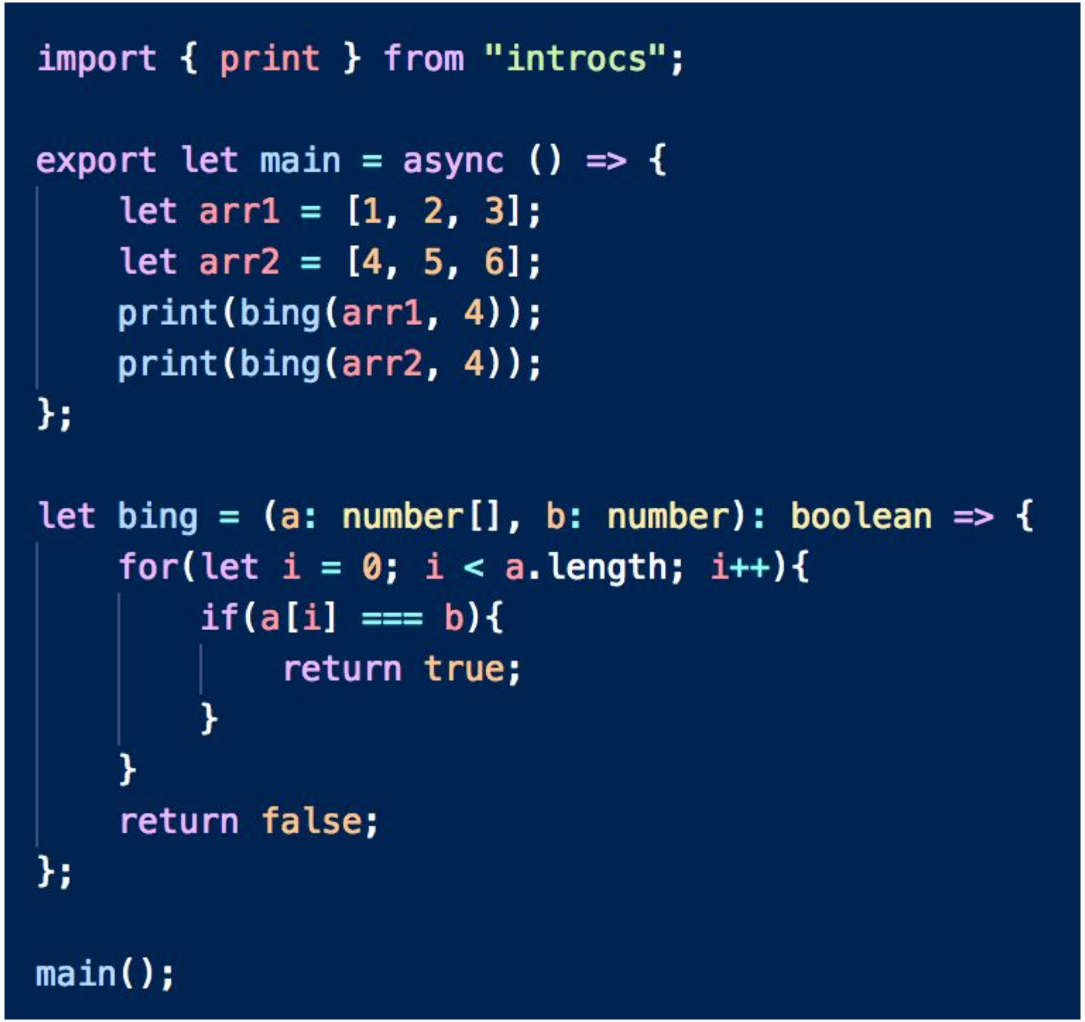
In what order does this code run?



```
export let main = async() => {

let seven =          ;
print(s   n);

};

let add = (x: number, y: number): number =>
    return    +
};

main();
```

Arguments, Parameters, and Returned Values

1. What are the arguments?
2. What are the parameters?
3. What is the return value?
4. What is the value of “seven”?



```
import { print } from "introcs";

export let main = async () => {
    let arr1 = [1, 2, 3];
    let arr2 = [4, 5, 6];
    print(bing(arr1, 4));
    print(bing(arr2, 4));
};

let bing = (a: number[], b: number): boolean => {
    for(let i = 0; i < a.length; i++){
        if(a[i] === b){
            return true;
        }
    }
    return false;
};

main();
```

Question 1:
What is printed after the following code executes?

Answer:
false
true

What does this function do?
What would be a better name for this function?

Check in on Course.Care

1811D

Scope

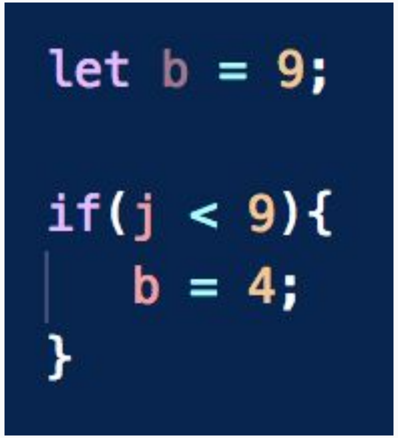
- Tracing the scope of a variable in TypeScript can be tricky, but we use two rules to define it.
- Firstly, the scope of a variable is defined by the innermost curly braces that surround its declaration
- Second, two variables cannot have the same name if their scopes overlap.

Let’s look at examples.

Scope

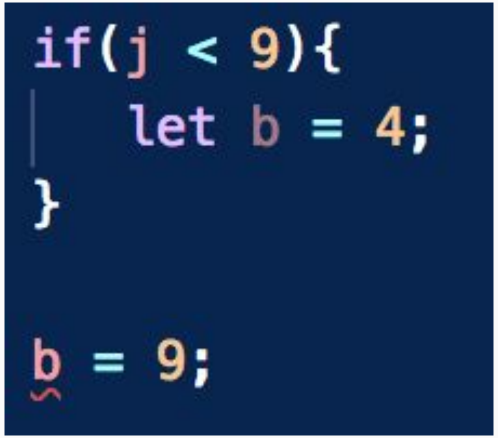
- The scope of a variable is defined by the innermost curly braces that surround its declaration. This can be thought of sort of like a 1-way mirror: information can always move inwards, but cannot travel outwards!

Scope



```
let b = 9;

if(j < 9){
    b = 4;
}
```

Valid scoping!

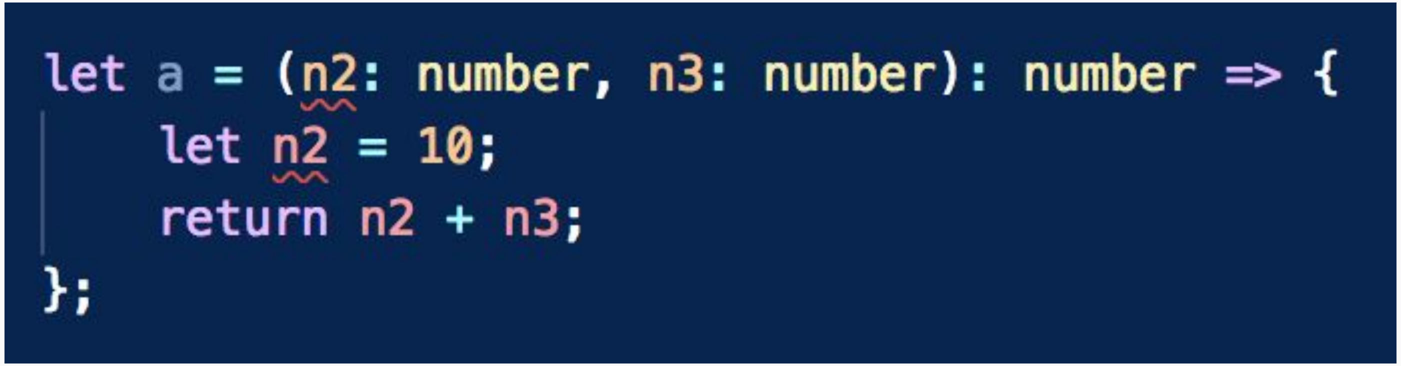


```
if(j < 9){
    let b = 4;
}

b = 9;
```

Invalid scoping!

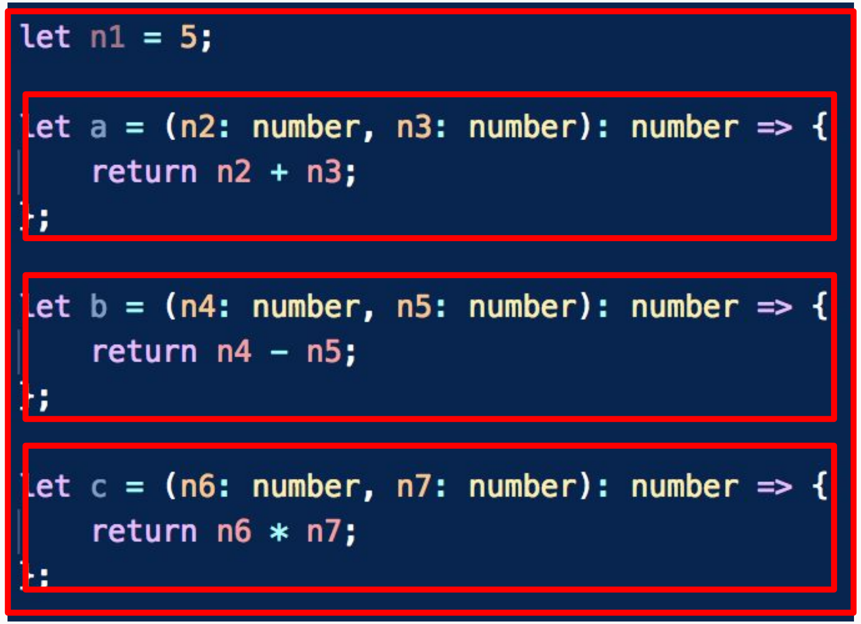
Scope

- Two variables cannot have the same name if their scopes overlap.
- Below, we see the variable “n2” declared twice in the same scope, eliciting a TS error.



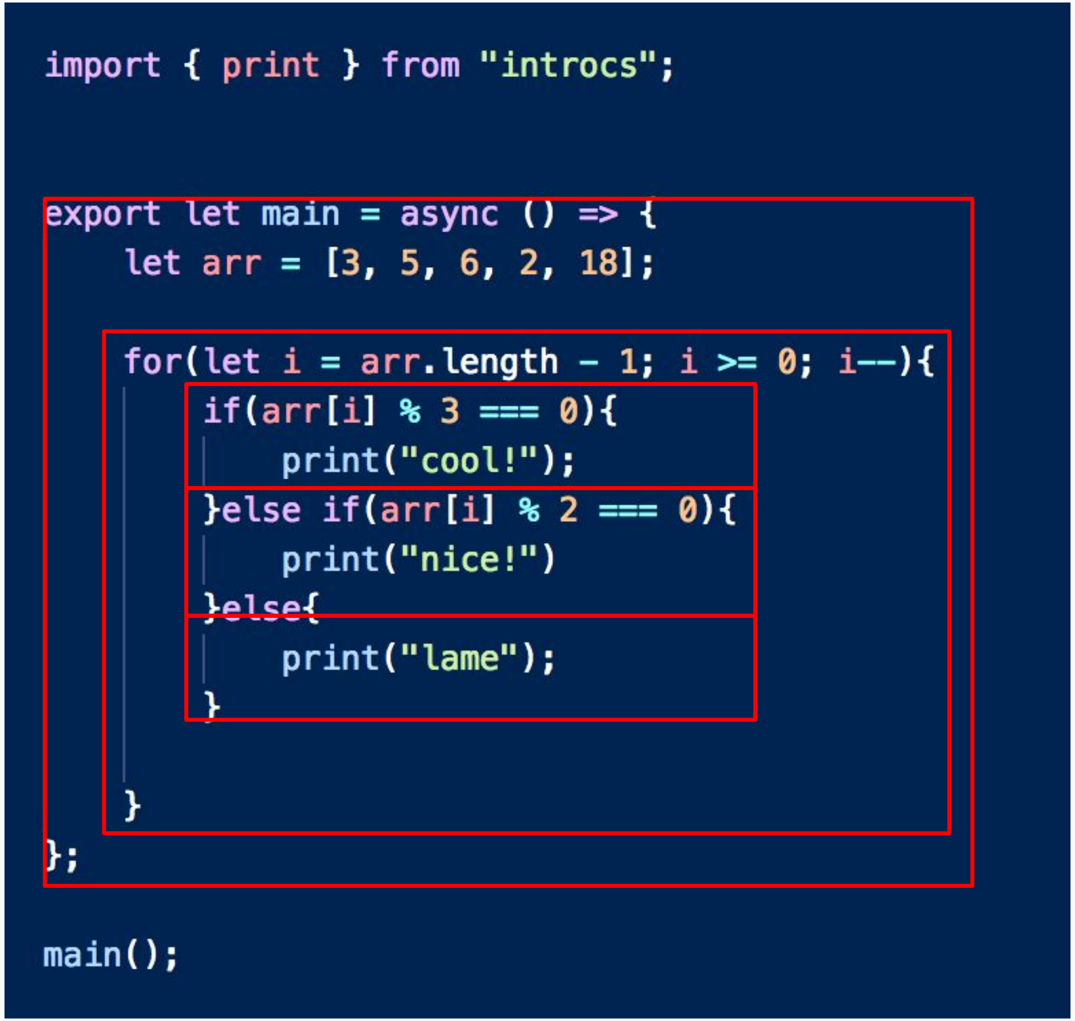
```
let a = (n2: number, n3: number): number => {
    let n2 = 10;
    return n2 + n3;
};
```

Scope



```
let n1 = 5;

let a = (n2: number, n3: number): number => {
    return n2 + n3;
};

let b = (n4: number, n5: number): number => {
    return n4 - n5;
};

let c = (n6: number, n7: number): number => {
    return n6 * n7;
};
```

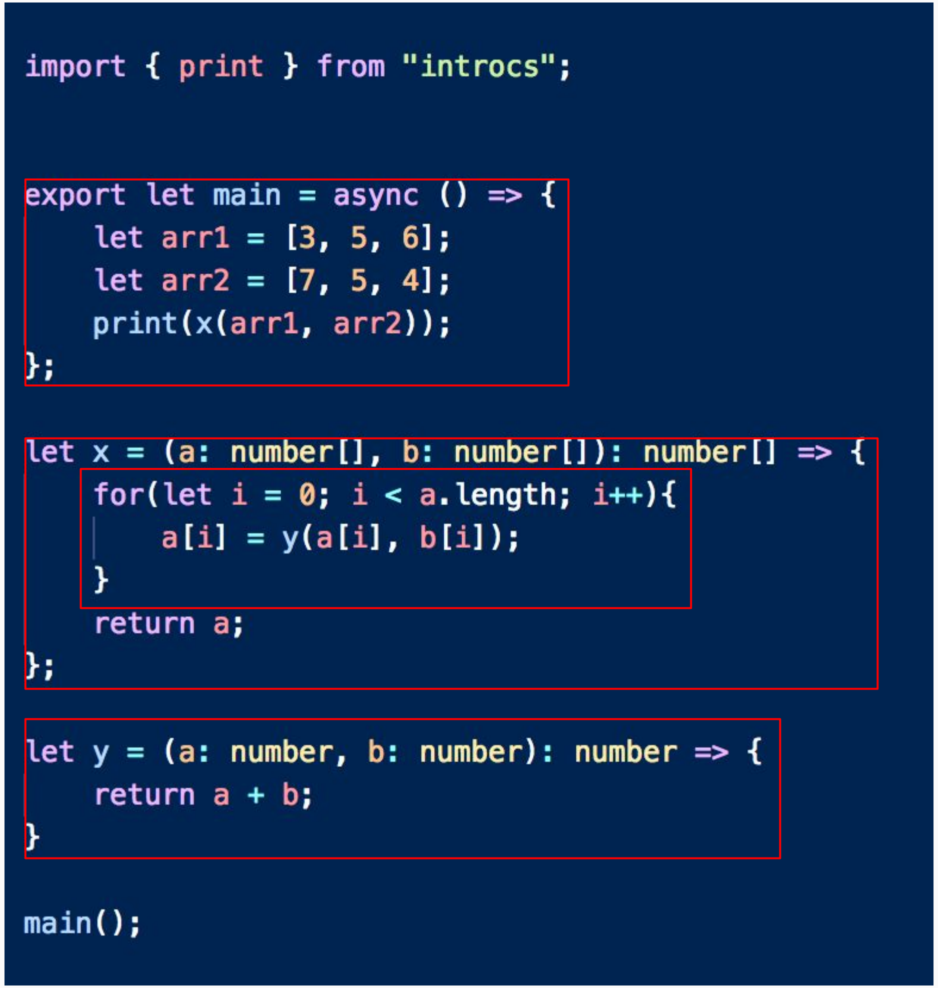
- The variable “n1” may be accessed anywhere in this code because it’s a global variable.
- The variables “n6” and “n7” may only be accessed within the function named “c”, and vice versa for the other variables.



```
import { print } from "introcs";


export let main = async () => {
    let arr = [3, 5, 6, 2, 18];

    for(let i = arr.length - 1; i >= 0; i--){
        if(arr[i] % 3 === 0){
            print("cool!");
        }else if(arr[i] % 2 === 0){
            print("nice!")
        }else{
            print("lame");
        }

    }
};

main();
```

Question 2:
What is printed as the following code executes?
What does it do?

Answer:
“cool!”
“nice!”
“cool!”
“lame”
“cool!”



```
import { print } from "introcs";


export let main = async () => {
    let arr1 = [3, 5, 6];
    let arr2 = [7, 5, 4];
    print(x(arr1, arr2));
};

let x = (a: number[], b: number[]): number[] => {
    for(let i = 0; i < a.length; i++){
        a[i] = y(a[i], b[i]);
    }
    return a;
};

let y = (a: number, b: number): number => {
    return a + b;
}

main();
```

Question 3:
What is printed as the following code executes?
What does the function do?

Answer:

[10, 10, 10]