```ts
function add(a,b) : number{
    return a + b;
}

function helper(a,b){
    console.log(a,b);
    let sum = add(a,b)
    console.log(a,b,sum)
}
helper(10,20)
```

10 20

10 20 30

30

where to return?

Callstack

```ts
function fibonacci(n: number): number {
    if (n == 1) return 0;
    if (n == 2) return 1;

    let fnm1 = fibonacci(n - 1);
    let fnm2 = fibonacci(n - 2);

    return fnm1 + fnm2;
}

let ans = fibonacci(5);
console.log(ans)
```

3

Factorial

$n! = 1 \times 2 \times 3 \times \dots \times n$

$n \Rightarrow n!$

```ts
function printer(n:number) : void{
    if(n == 0){
        return;
    }
    console.log(n);
    printer(n-1);
    console.log(n);
}

printer(5)
```

- Call → stack pe add
- Return → remove from stack
  - Where? Jaha se call hua
- All lines complete
- Top → execute

Call Stack

- Stack top → execution happening

or

5 4 3 2 1

1 2 3 4 5

- Call Stack - Understand
- Components - Code
- Fair expectation logic?

Fibbonacci Numbers $f(n) = f(n-1) + f(n-2)$

n	0	1	2	3	4	5	6	7	8	9	10
f(n)	0	1	1	2	3	5	8	13	21	34	55

n^{th} fib. no.?

```ts
function findFirstIdx(idx:number,target:number,arr:number[]) : number{
    if(arr[idx] == target) return idx;
    let ans = findFirstIdx(idx+1,target,arr);
    return ans;
}
```

idx = 0

target = 1

if(idx == arr.length) return -1

- Last idx
- Stack for current idx, last occurrence of target

- Fair
- expected

Guarantee — Faith $f(4)$ ✓

1. Parameter → n
2. Ret. type → number

Tree