* Currying in javascript (interview question)

→ currying is a technique of evaluating function with multiple arguments, into sequence of function with single argument.

→ In other words, when a function, instead of taking all arguments at one time; takes the first one and return a new function that takes the second one and returns a new function which takes the third one, and so forth, until all arguments have been fulfilled.

→ Now we will see two ways to curry functions
① using bind function
② using closure concept

① using bind function

```
let multiply = function (x,y) {
    console.log(x*y);
}
let multiply Two = multiply.bind(this, 2);
multiply Two (5);
let multiply Three = multiply.bind (this,3);
multiply Three (5);
```

↓
10
15

[here x is 2
and y is 5
in multiply Two
method

This is called as currying in javascript.
Here we are using multiply method to create
multiplyTwo and multiplyThree method.

→ Now what if we pass one more argument

```
let multiply = function(x,y) {
     console.log(x*y);
}
let multiplyTwo = multiply.bind(this,2,3);
multiplyTwo (5);
```

[x will be 2]
[y will be 3]

↓

6

so here 3 will end up with "y" and 5 will
be ignored which is in multiplyTwo method.

→ Now what if we pass both arguments using
multiplyTwo method, then also it will work.

```
let multiply = function(x,y) {
     console.log(x*y);
}
let multiplyTwo = multiply.bind(this,);
multiplyTwo (4,5);
```

[x will be 4]
[y will be 5]

↓

20

② Currying using concept of closures

```
let multiply = function(x){
    return function(y){
        console.log(x*y);
    }
}
let multiplyTwo = multiply(2);
multiplyTwo(3);
```

this function also has access to x due to closures

↓

6

→ so these are two ways to curry functions.

→ currying refers to the process of transforming a function with multiple arity into the same function with less arity. The curried effect is achieved by binding some of the arguments to the first function invoke, so that those values are fixed for the next invocation.

→ Arity
arity is the term used to refer to the number of arguments or operands in a function or operation, respectively.