JavaScript Recusion Review

- Recursion is a programming concept that allows you to call a function repeatedly until a base-case is reached.

Here is an example of a recursive function that calculates the factorial of a number:

Example Code

```
function findFactorial(n) {
 if (n === 0) {
  return 1;
 }
 return n * findFactorial(n - 1);
}
```

In the above example, the findFactorial function is called recursively until n reaches 0. When n is 0, the base case is reached and the function returns 1. The function then returns the product of n and the result of the recursive call to findFactorial(n - 1).

- Recursion allows you to handle something with an unknown depth, such as deeply nested objects/arrays, or a file tree.
- A call stack is used to keep track of the function calls in a recursive function. Each time a function is called, it is added to the call stack. When the base case is reached, the function calls are removed from the stack.
- You should carefully define the base case as calling it indefinitely can cause your code to crash. This is because the recursion keeps piling more and more function calls till the system runs out of memory.
- Recursions find their uses in solving mathematical problems like factorial and Fibonacci, traversing trees and graphs, generating permutations and combinations and much more.