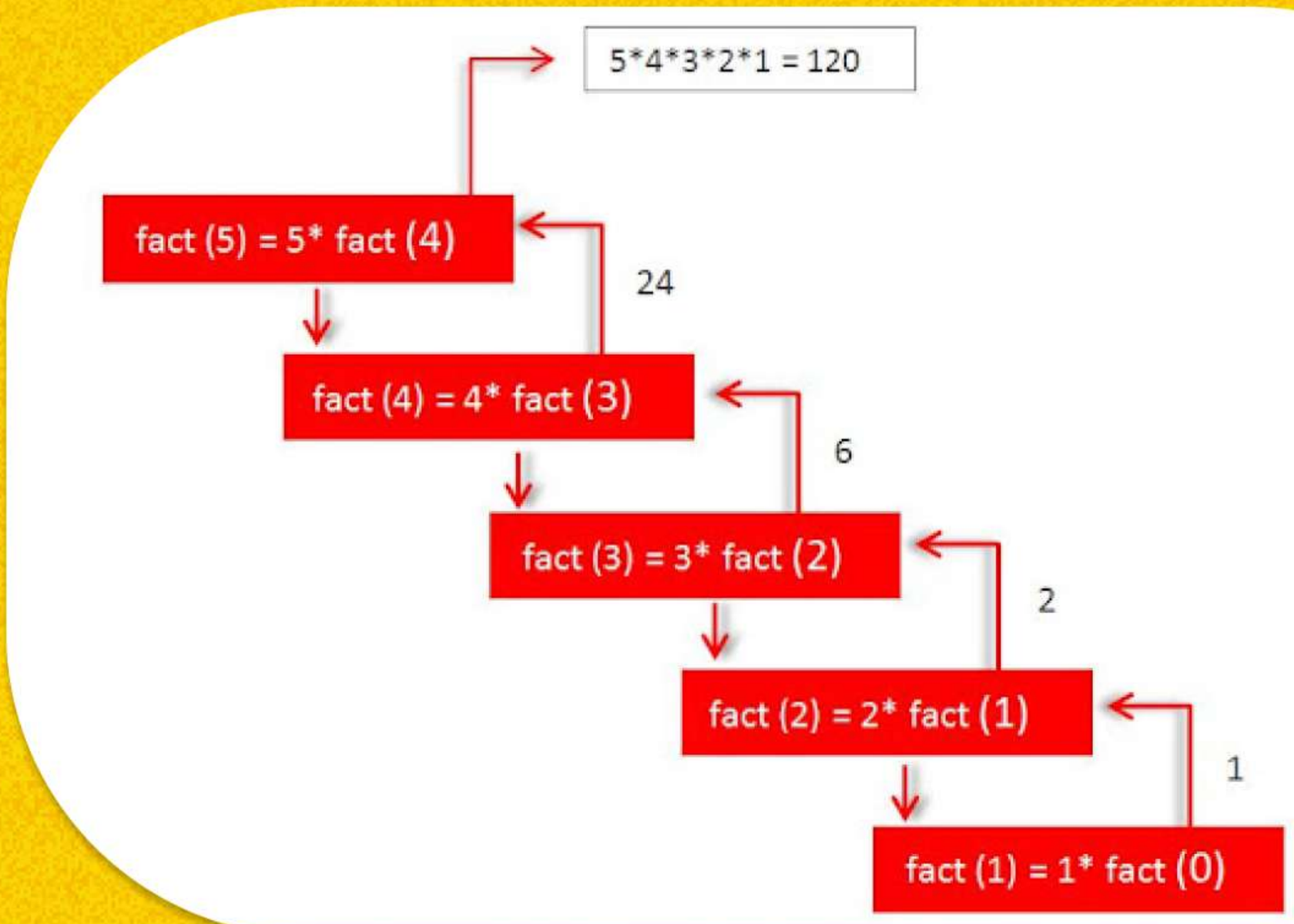
# Understanding Recursion in JavaScript

Sayyed Siddique

swipe ›

# 1 Introduction to Recursion

Recursion is a process in which a function calls itself as a subroutine. This technique is essential in solving problems that can be broken down into smaller, repetitive tasks.




Sayyed Siddique

swipe ›

# 2 Basic Structure of Recursion

A recursive function typically has two main parts:

**Base Case:** The condition under which the function stops calling itself.

**Recursive Case:** The part where the function calls itself with a different argument.

```
function recursiveFunction(args) {
  if (baseCaseCondition) {
    // base case logic
  } else {
    // recursive case logic
    recursiveFunction(newArgs);
  }
}
```

Sayyed Siddique

swipe ›

# 3 Example: Factorial of a Number

The factorial of a number nnn is the product of all positive integers less than or equal to nnn. It can be defined recursively as:

```
function factorial(n) {
  if (n === 0) {
    return 1; // base case
  }
  return n * factorial(n - 1); // recursive case
}
console.log(factorial(5)); // Output: 120
```

Sayyed Siddique

swipe ›

# 4 Visualizing the Process

Let's visualize how the factorial function works when calculating factorial(3):

1. Call stack: factorial(3)
2. Call stack: factorial(3) -> factorial(2)
3. Call stack: factorial(3) -> factorial(2) -> factorial(1)
4. Call stack: factorial(3) -> factorial(2) -> factorial(1) -> factorial(0)
5. Resolve from base case up: 1 -> 1 * 1 -> 2 * 1 -> 3 * 2

Sayyed Siddique

swipe ›

# 5 Practical Uses of Recursion

Recursion is useful in

1. **Tree and Graph Traversal:** Exploring nodes in a tree or graph structure.
2. **Sorting Algorithms:** Such as QuickSort and MergeSort.
3. **Dynamic Programming:** Solving complex problems by breaking them down into simpler subproblems.

Sayyed Siddique

swipe ›

# Sayyed Siddique

“ Please like; if you find value,
consider following! ”