JavaScript

Recursion

For something simple to start with – let’s write a function pow(x, n) that raises x to a natural power of n. In other words, multiplies x by itself n times.

pow(2, 2) = 4

pow(2, 3) = 8

pow(2, 4) = 16

**var 1**

function pow(x, n) {

let result = 1;

// multiply result by x n times in the loop

for (let i = 0; i < n; i++) {

result \*= x;

}

return result;

}

alert( pow(2, 3) ); // 8

**var 2 Recursive thinking: simplify the task and call self:**

**check how n changes on each pow call**

function pow(x, n) {

if (n == 1) {

return x;

} else {

return x \* pow(x, n - 1);

}

}

alert( pow(2, 3) ); // 8

## **[Recursive traversals](https://javascript.info/recursion" \l "recursive-traversals)**

Tasks

### **[Sum all numbers till the given one](https://javascript.info/recursion" \l "sum-all-numbers-till-the-given-one)**

function sumTo(n) {

if (n == 1) return 1;

return n + sumTo(n - 1);

}

alert( sumTo(100) );

1. The [factorial](https://en.wikipedia.org/wiki/Factorial) of a natural number is a number multiplied by "number minus one", then by "number minus two", and so on till 1. The factorial of n is denoted as n!

function factorial(n) {

return n ? n \* factorial(n - 1) : 1;

}

alert( factorial(5) ); // 120

1. The sequence of [Fibonacci numbers](https://en.wikipedia.org/wiki/Fibonacci_number)

has the formula Fn = Fn-1 + Fn-2. In other words, the next number is a sum of the two preceding ones.

First two numbers are 1, then 2(1+1), then 3(1+2), 5(2+3) and so on: 1, 1, 2, 3, 5, 8, 13, 21....

Fibonacci numbers are related to the [Golden ratio](https://en.wikipedia.org/wiki/Golden_ratio) and many natural phenomena around us.

Write a function fib(n) that returns the n-th Fibonacci number.

function fib(n) {

return n <= 1 ? n : fib(n - 1) + fib(n - 2);

}