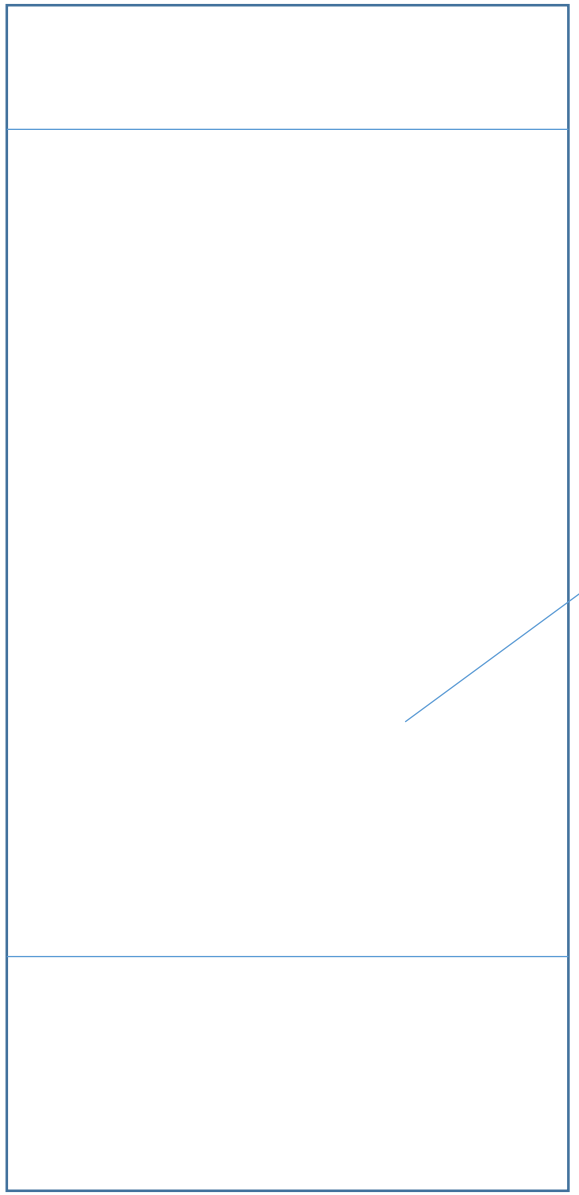
How recursion.js works

By John Diyala

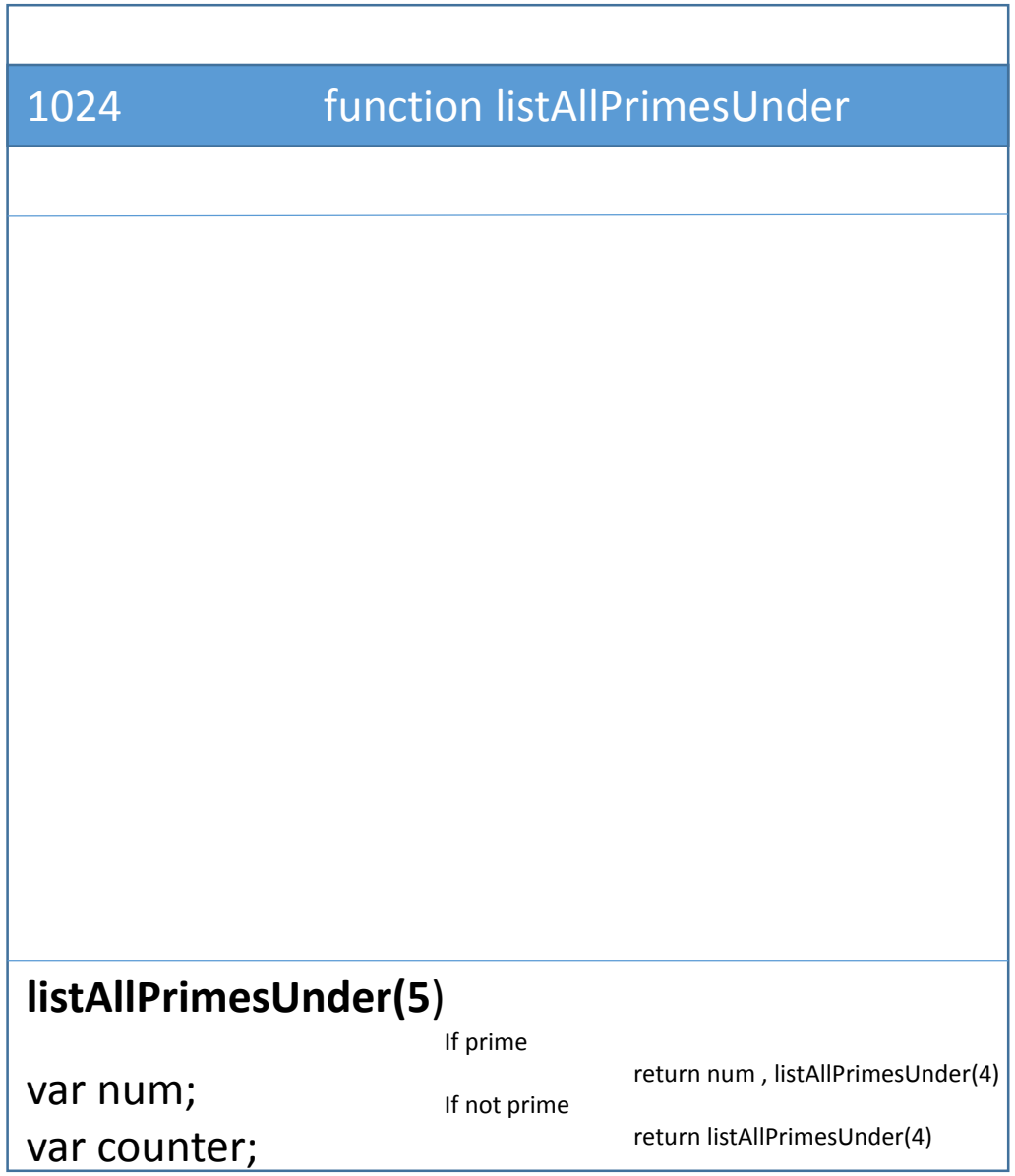
*pseudo-code was used for a simpler explanation



Heap Memory

Available memory (can be used as stack or heap memory as needed)

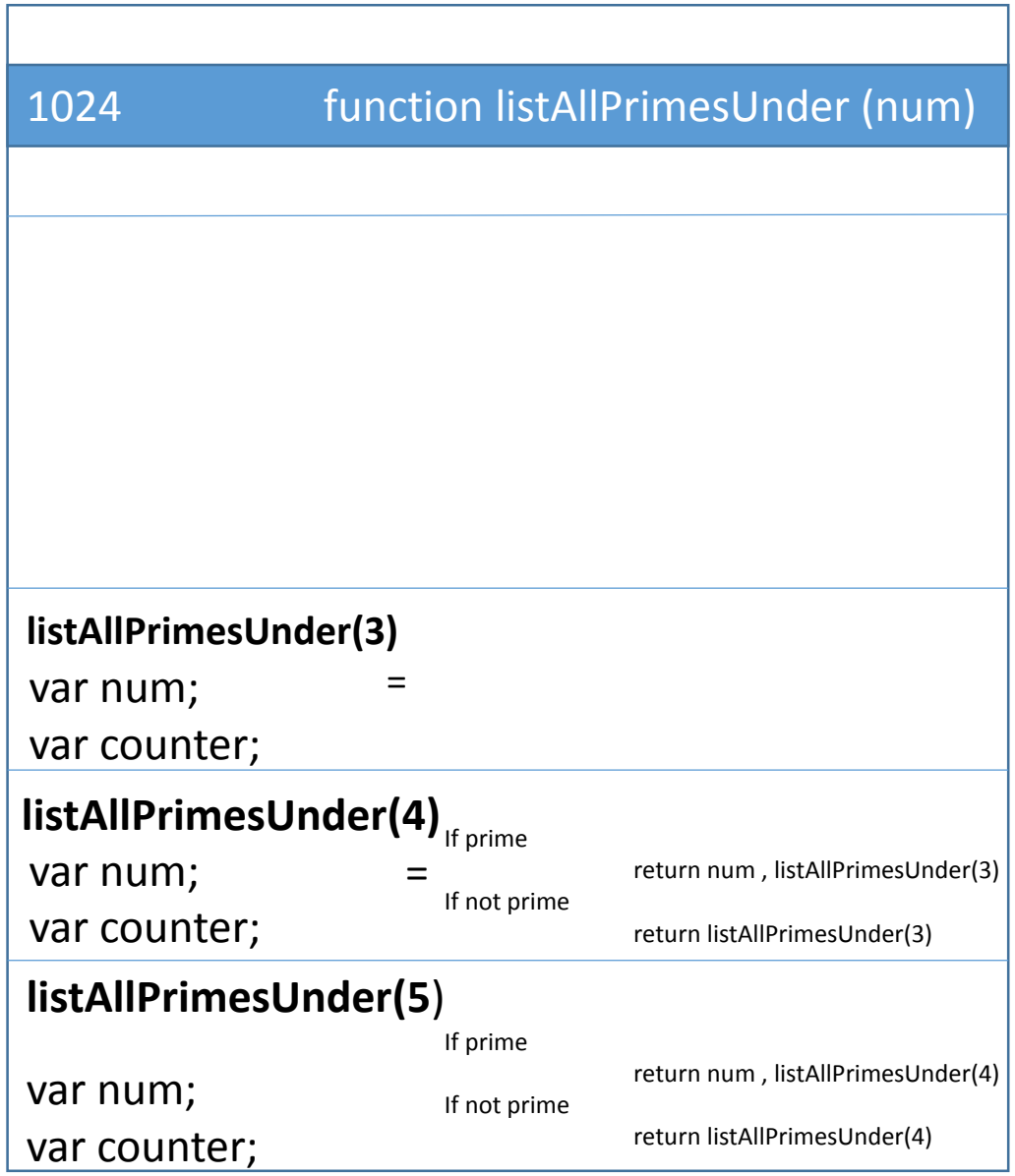
Stack Memory



A pointer to the function is stored on the heap

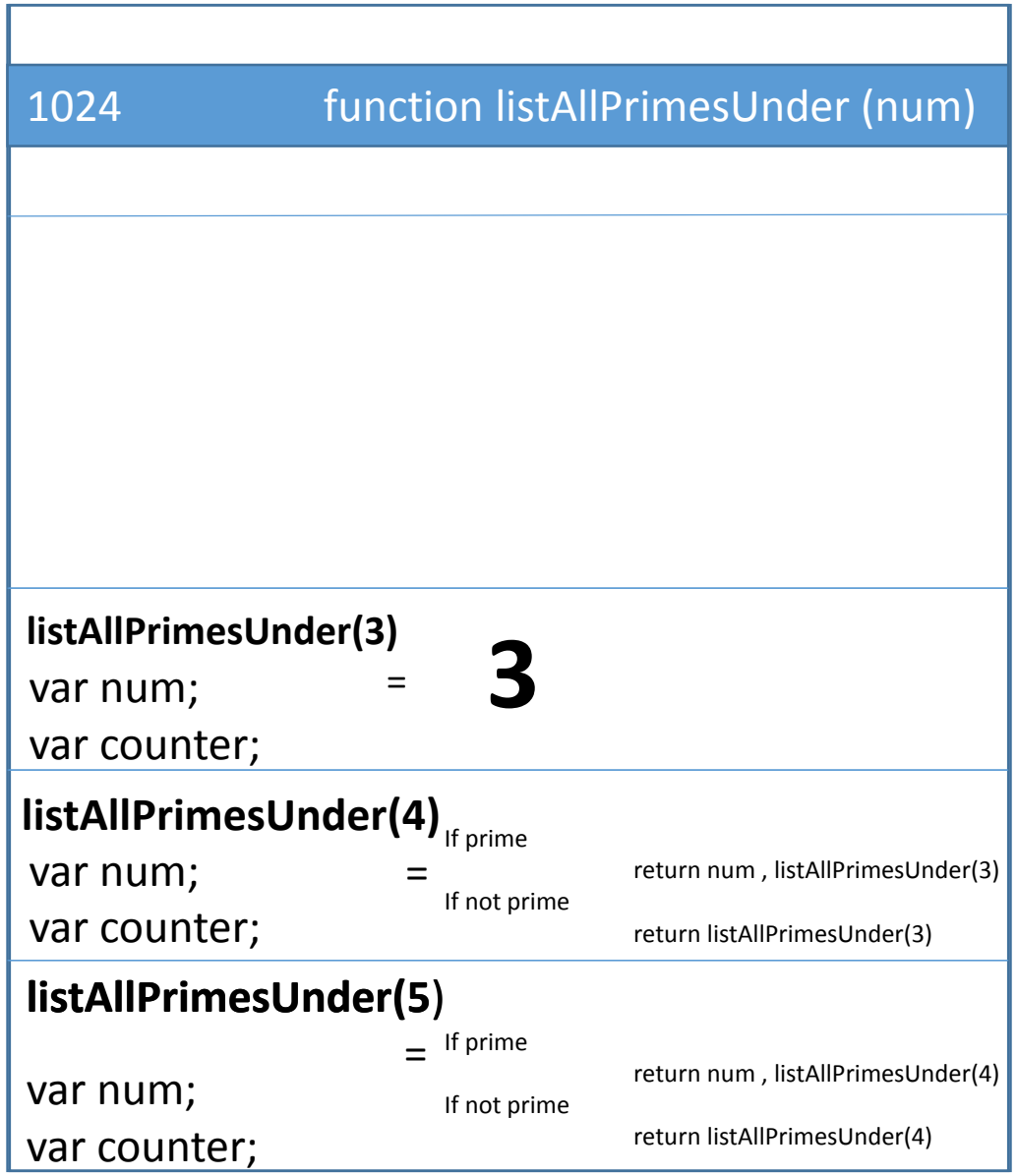
When listAllPrimesUnder(5) is called, Javascript allocates a part of the stack, and tries to evaluate the answer.

All the variables within listAllPrimesUnder(5) are stored on the stack -in this case there only exists the num variable (5) and the counter variable that is used to count the number of numbers that divide evenly into 'num', in order for num to be a prime, this number should be 0 (1 and num itself are not counted)



Since listAllPrimesUnder(5) needs to call = listAllPrimesUnder(4), another call is added to the stack frame.

Consequently, when listAllPrimesUnder(4) executes, listAllPrimesUnder(3) is also added to the stack.



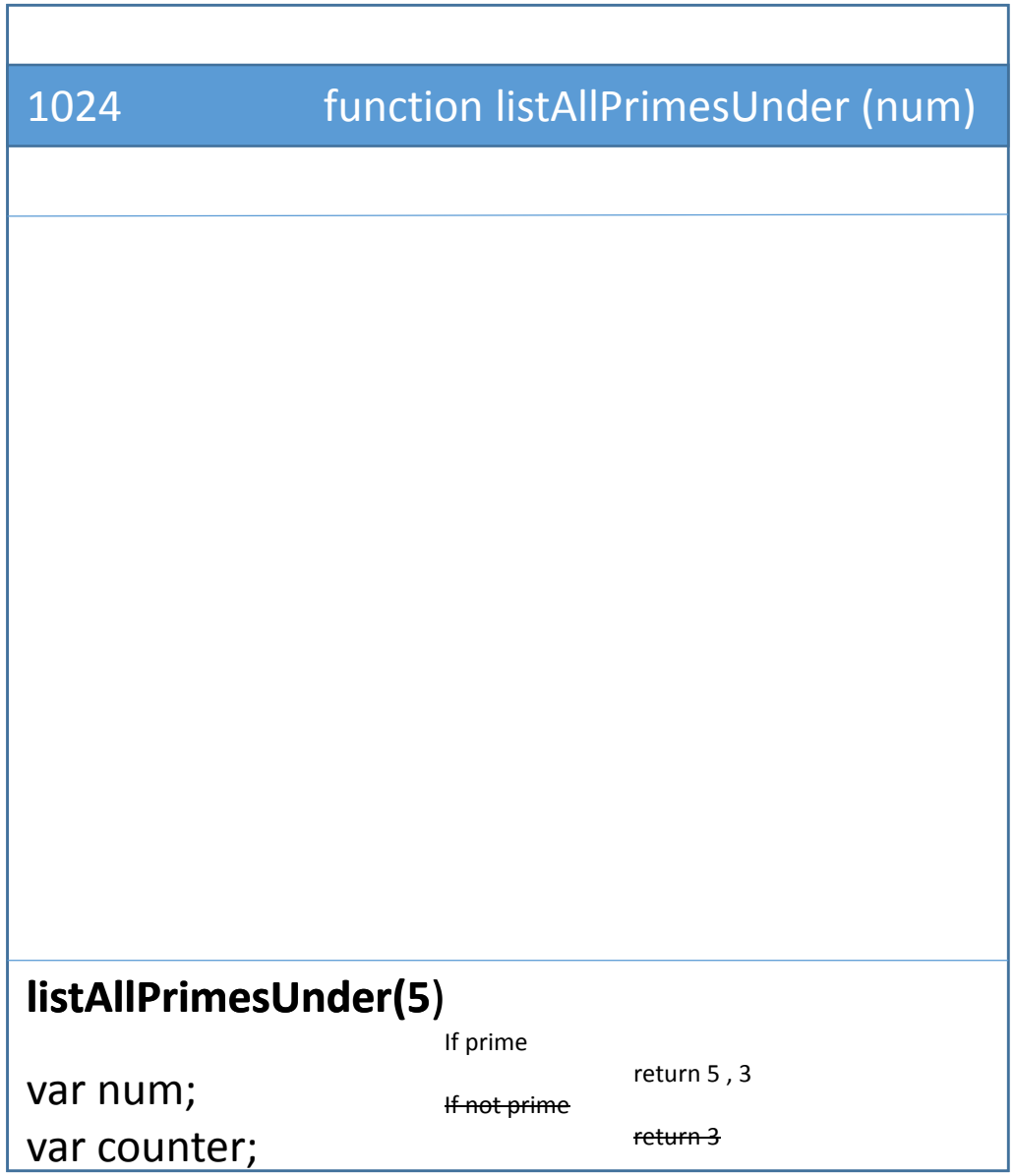
Since listAllPrimesUnder(3) is the base case, and returns 3

It is popped off the stack and the value (3) is returned to listAllPrimesUnder(4)

1024	function listAllPrimesUnder (num)
listAllPrimesUnder(4) var num; var counter;	= **3**
listAllPrimesUnder(5) var num; var counter;	If prime return 5, 3 If not prime return listAllPrimesUnder(4)

listAllPrimesUnder(4) can now execute since it has received a value for listAllPrimesUnder(3):

Since 4 is not a prime, listAllPrimesUnder(4) does not add 4 to the list of primes (that only contains 3 at the moment).



When 4 is also popped off listAllPrimesUnder(5) evaluates to

5, 3

R1.3)

- Describe what modifications to the stack and heap model provided, if any, are necessary to convey how recursion works with javascript functions. That is, can you illustrate your recursion using the model provided or does it fall short and require modifications or new features?

- I didn't provide any modifications to the stack and heap model provided.
  - Since the heap is used for storing references to dynamic data types the only thing that is stored on the heap in this example the function "listAllPrimesUnder(num)".
  - Each and every time listAllPrimesUnder(num) is called with a different number, a new call is added to the call stack, and the two variables (num, counter) are kept in that stack.