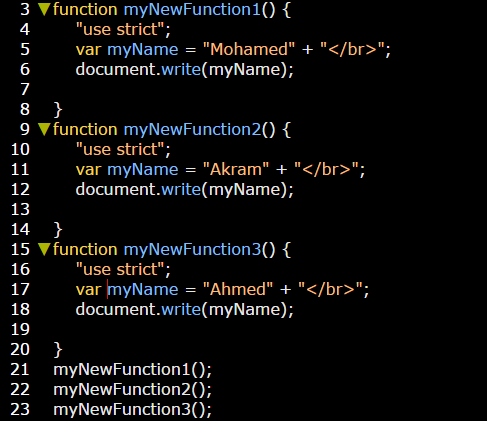
**Call Stack**

**Stack**: is a part of memory used to keep track of function calls.

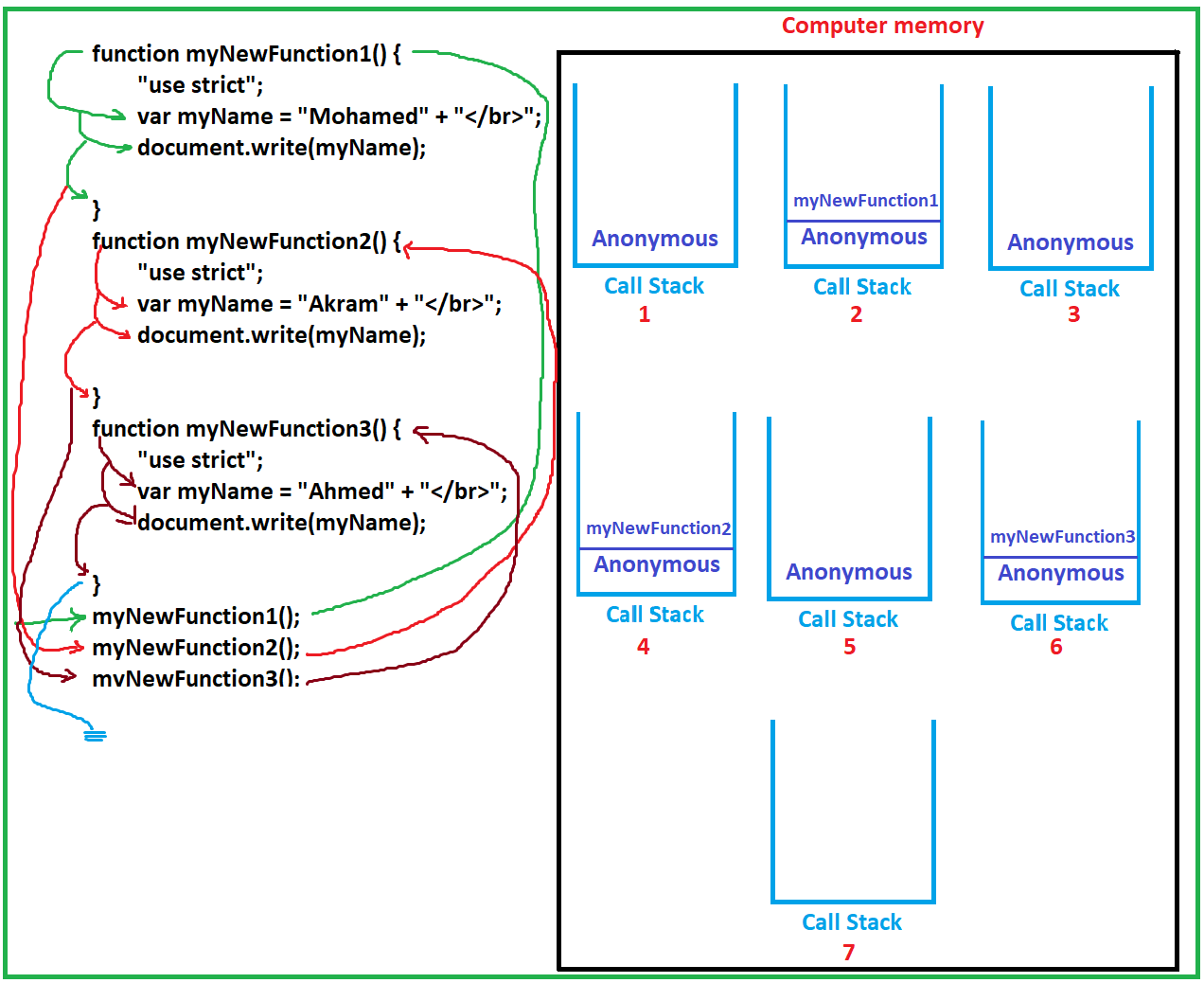
When we call a function, that function is going to be **pushed on** top of the stack & Local variables of that function are **created** in the local scope.

While when the control exits a function it is going to be **popped out** of the stack & the created local variables of that function are **deleted** from the local scope

**Example:**



**How this statement works?**



1. As soon as the functions created in your js script a specific space in your computer memory called “Call Stack” was preserved.
2. Call Stack memory used to organize and track the process of calling these functions to execute them.
3. As soon as Call Stack memory be preserved, a default function stored on it. This function is an anonymous function **(1)**.
4. When the cursor starting execute function call, the last one was stored above the previous anonymous inside the call stack memory **(2)**.
5. After the cursor go inside this function, variables inside it were created in the local scope of this function. See scope option in source tool
6. The cursor starts to execute function body.
7. After that the cursor goes out function body, and the variables inside will deleted from the local scope of this function.
8. After execute all above steps the current function itself, will be deleted from Call Stack memory **(3)**.
9. The next function will be stored in Call Stack memory **(4)**, and passes through all above steps.... these steps will repeated will all functions that needed to call.
10. After calling all current functions the Anonymous function will be deleted from call stack **(7)**, the last became completely empty; that indicates all functions were executed.
11. The cursor goes out these functions to execute down statements.[

\]