**Scope in JavaScript**

Scope in JavaScript indicates to where variables (var, const, and let) live. It used to indicate to the **accessibility** and **visibility** of these variables.

There are three types of scopes in JavaScript:

1. Local or function scope.
2. Block scope
3. Global scope.

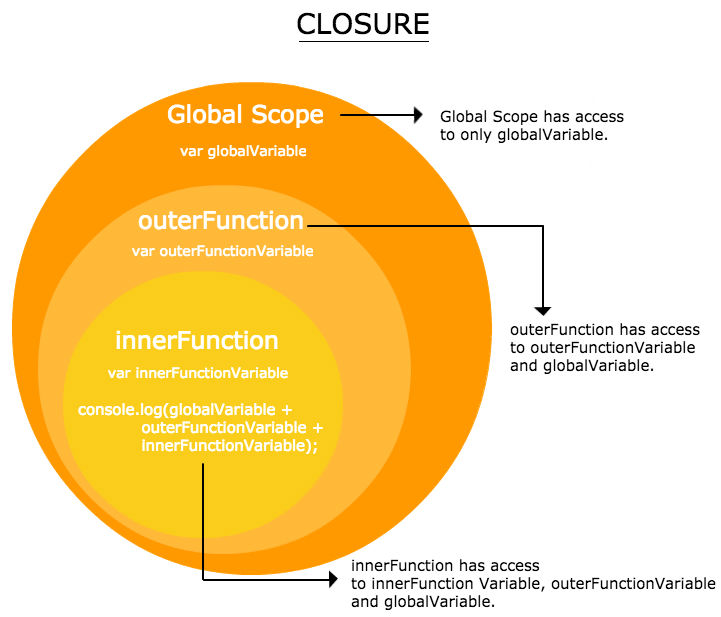
|  |  |  |
| --- | --- | --- |
| **Local Scope (function scope)** | **Block scope** | **Global Scope (all Script without function body)** |
| Local scope referee to [function body](Functions.docx) area. | Block scope referee to block [conditional statement](Conditional%20statement.docx) area. | Global scope referee to all script except function & block scope. |
| Variable inside local scope called local variables. | Variable inside block scope called block variables. | Variable inside global scope called global variables. |
| To call these variables we must put output statements **inside function body**. |  | To call these variables we put output statements **inside anywhere** in this script. |
| Both variable declaration and output statement **must** be located inside the function body. |  | **Both** variable declaration and output statement located **mustn’t** be inside the local scope. |
| Local variables are created as soon as the control enters the function body and deleted as soon as the control exits the function body. |  |  |
|  |  |  |
| Local & block scope locate inside global scope so they consider as **a part** of the global scope | | |

**Note:** Local variables can be converted to global variable by removing variable keyword. See example 7

**Nested scope:**

Nested scope produces when several functions became nested, where each function has its private scope.

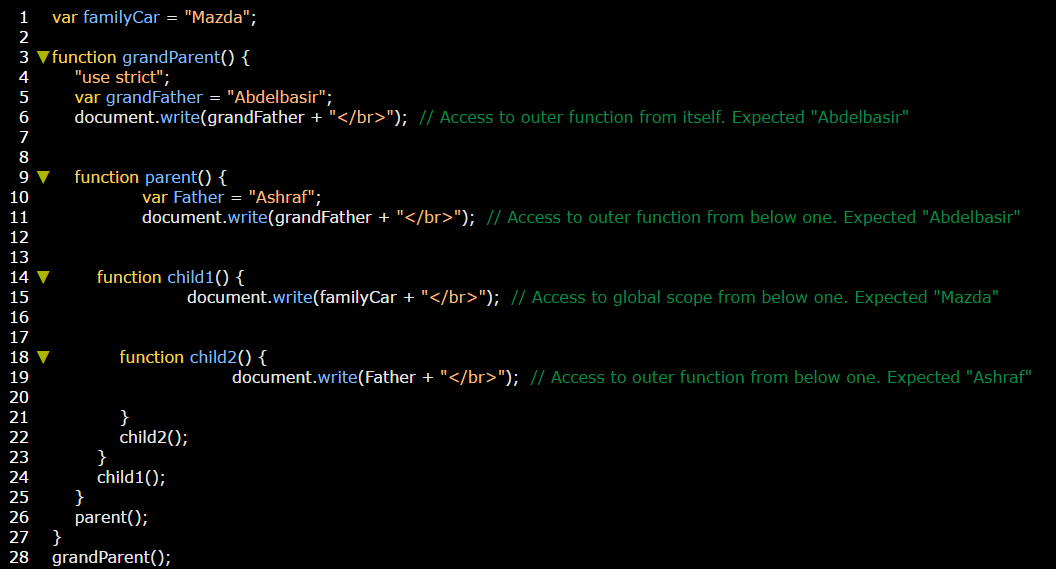
This picture illustrates Nested functions:



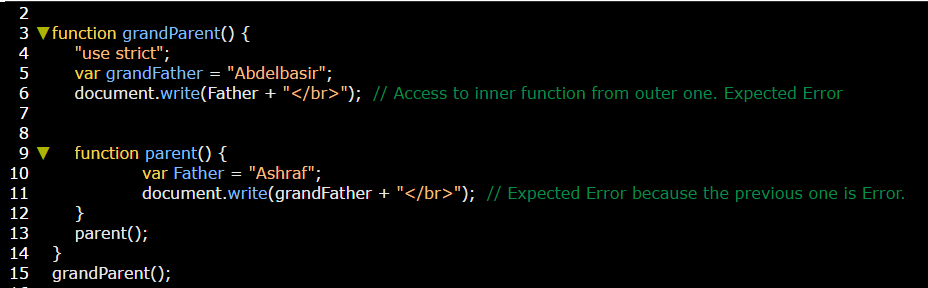
**Note:**

* All functions have access to the **global** scope.
* In fact, in JavaScript, all functions have access to the scope "**above**" them. In other words Variables that locate inside a specific function in nested scope can be accessed by putting output statement inside the **current** function or inside the **outer** one, it’s can be accessible from the inner function.
* All function can’t access to inner functions.

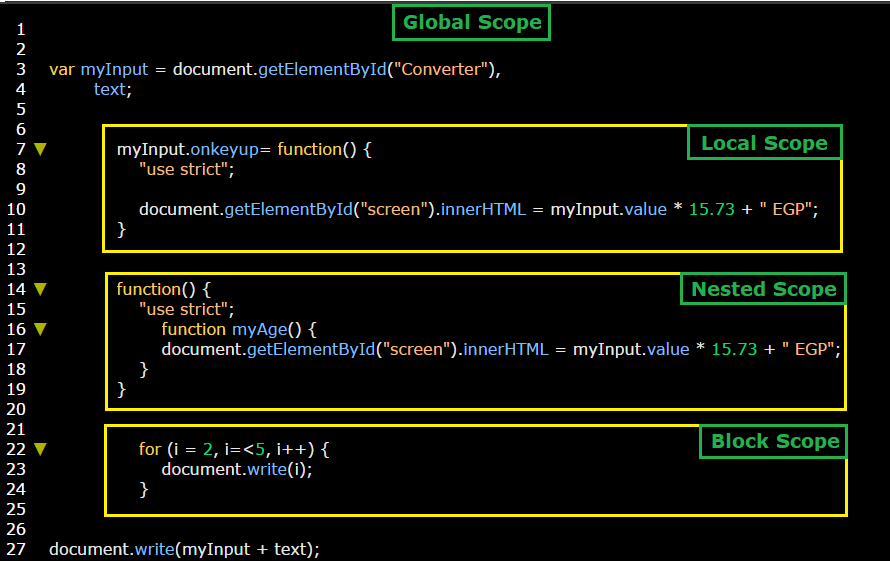
**Example 1:** on access to the global & outer function from below one.



**Example 2:** All function can’t access to inner functions.



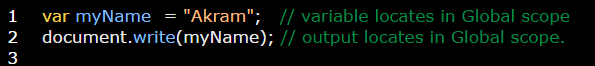
This screen illustrates the difference between local, Block, and global scope.



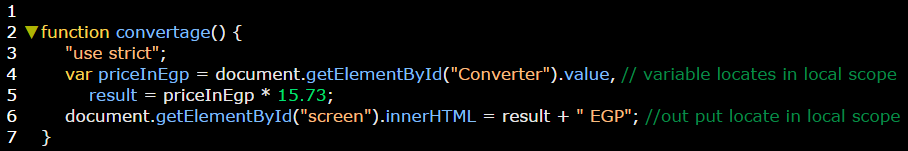
**Summary:**

|  |  |  |
| --- | --- | --- |
| Variable scope | Output statement scope | result |
| **global** | **global** | Print statement |
| **local** | **local** | Print statement |
| **block** | **block** | Print statement |
| **block** | **global** | Error |
| **local** | **global** | Error |
| **global** | **local** | Print statement |
| **Outer Nested function** | **The same function** | Print statement |
| **Outer Nested function** | **inner Nested function** | Error |
| **Outer Nested function** | **global** | Print statement |
| **inner Nested function** | **outer Nested function** | Print statement |
| **inner Nested function** | **global** | Print statement |
| **inner Nested function** | **The same function** | Print statement |
| **global** | **below Nested function** | Error |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

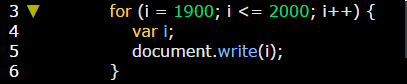
**Example 1:** global variable + global output statement = print statement

****

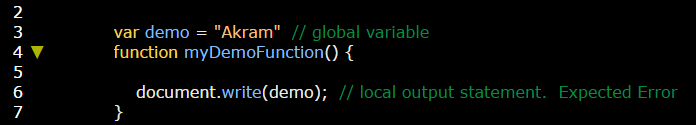
**Example 2:** local variable + local output statement = print statement

****

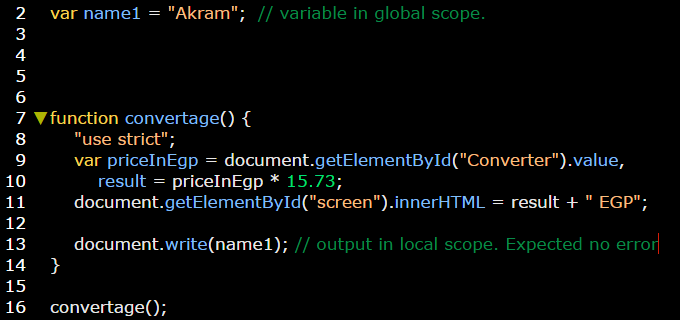
**Example 3:** block variable + block output statement = print statement

****

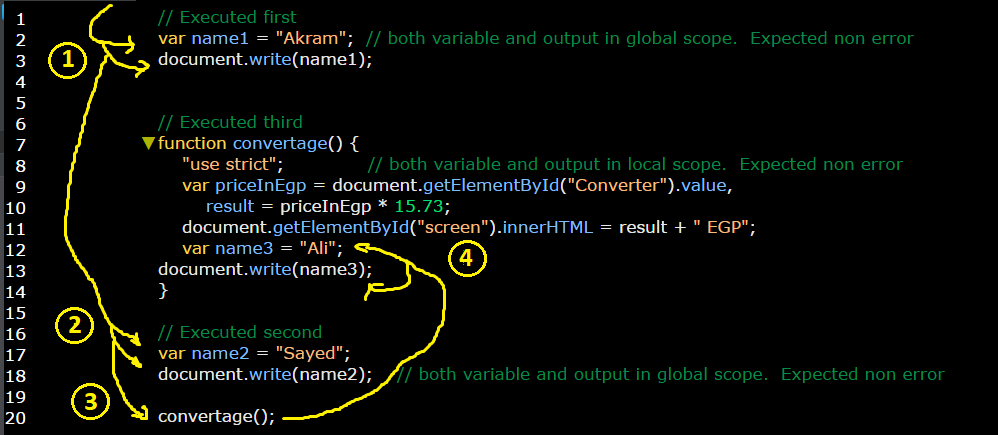
**Example 4:** variable **local** but output **global** >>> Error

****

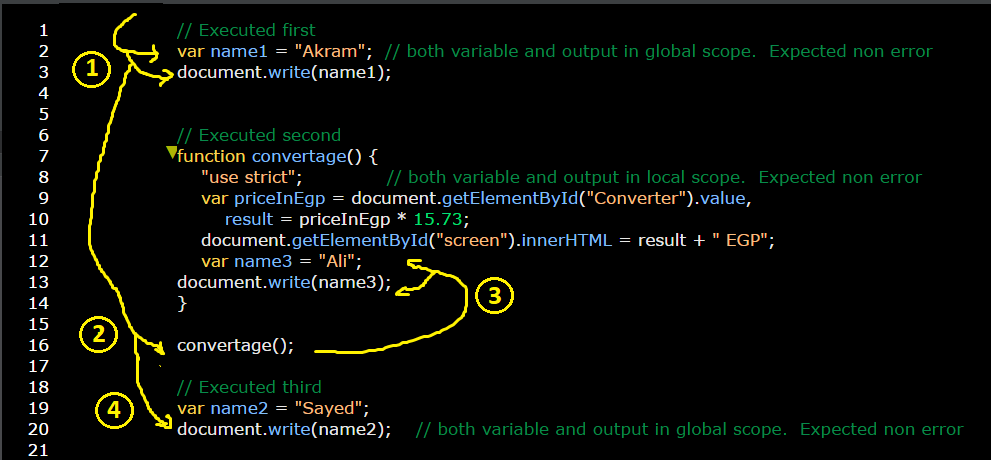
**Example 4:** variable **global** but output **local** >>> no Error



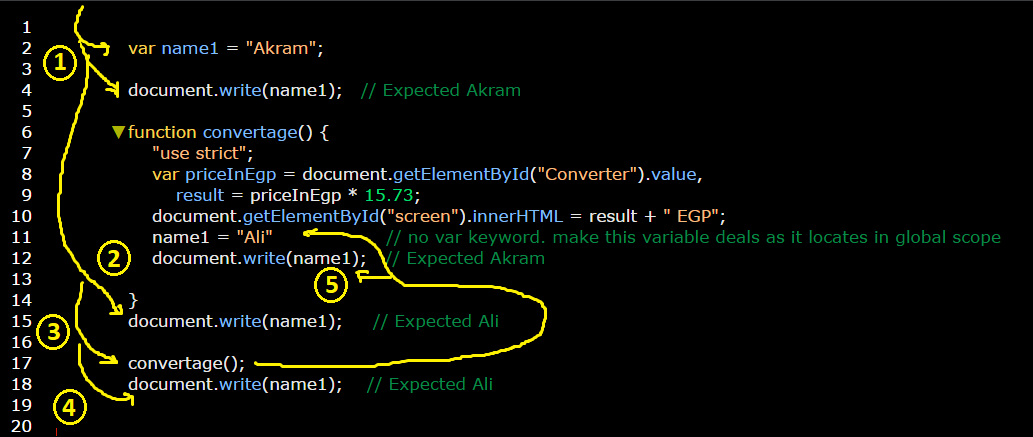
**Example 5:** Path of executing the variables in local and global scopes. Global > global > local



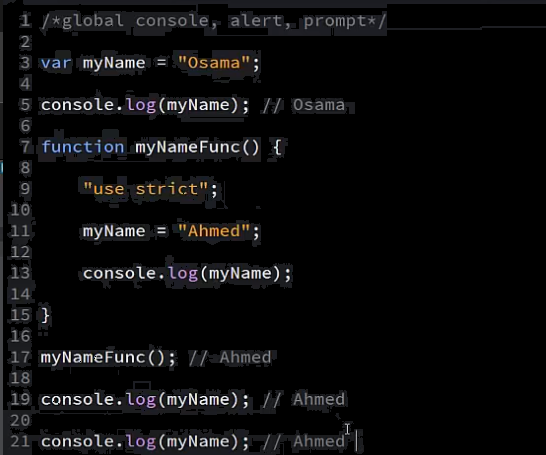
**Example 6:** Path of executing the variables in local and global scopes. Global > local > global



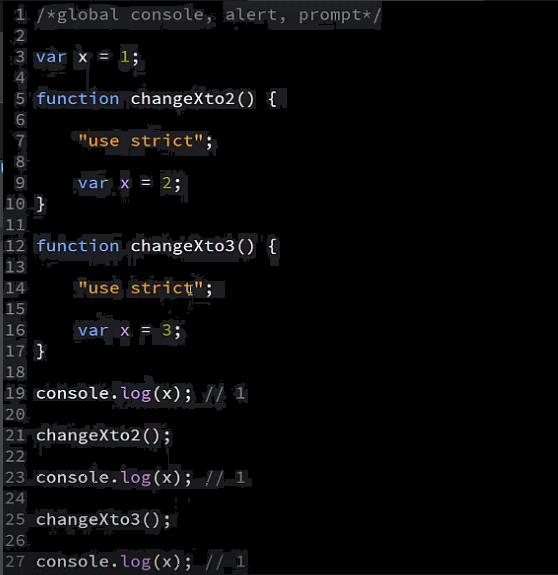
**Example 7:** convert local scope variable to global scope variable by removing variable keyword.



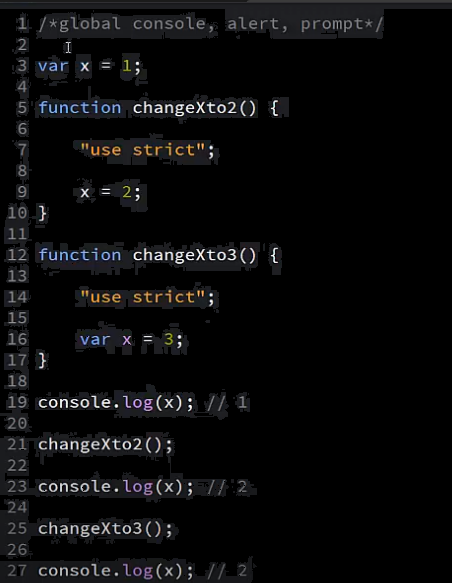
**Example 8:** Another example on converting local variable to global variable by removing variable keyword.



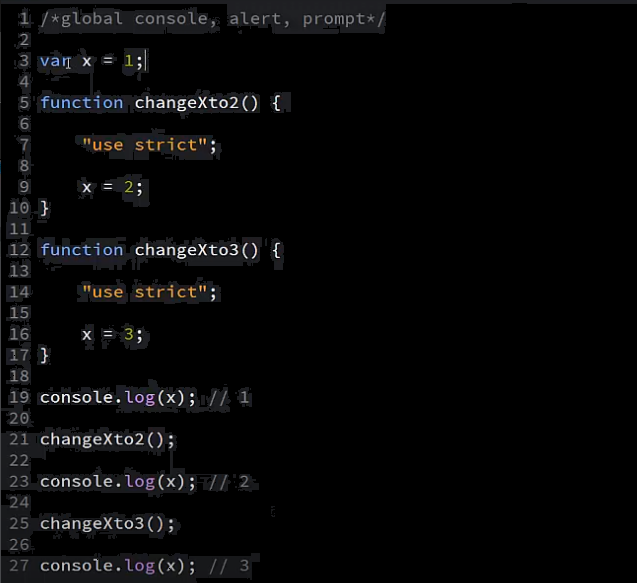
**Example 9:** Another example on converting local variable to global variable by removing variable keyword.



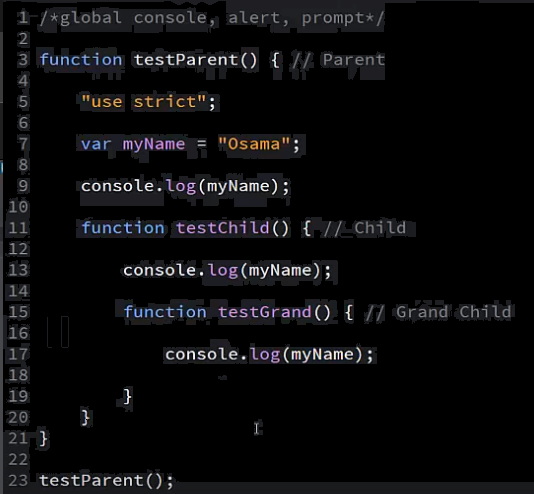
**Example 10:** Another example on converting local variable to global variable by removing variable keyword.



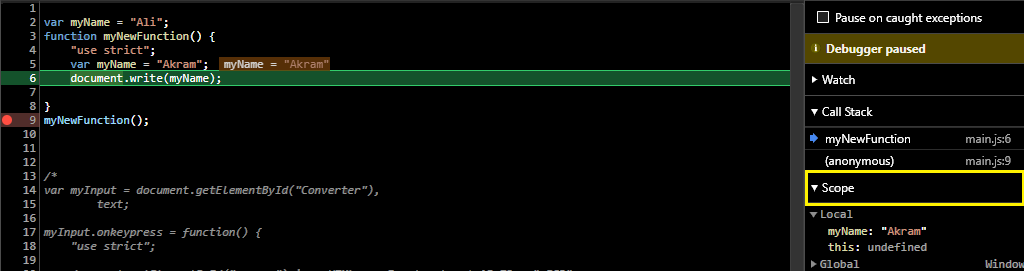
**Example 11:** Another example on converting local variable to global variable by removing variable keyword.



**Example 12:** example on local and global scope using nested functions.



**You can see the scope list from source tool**



**Note:** the best practice of using scope variables is use local and function, and block scopes and avoid as you can global scope, because if you have more than JS file in your page overriding and confusion will occur led to some problems.