**Week 07**

[**Variable Scope, Closure**](https://javascript.info/closure)

Scope is the accessibility of variables, functions or objects in a piece of code when it is being executed, that is, it is about visibility, about getting access. It can be accessible/visible in every script (global scope) or only in a certain section, function or block (local scope)

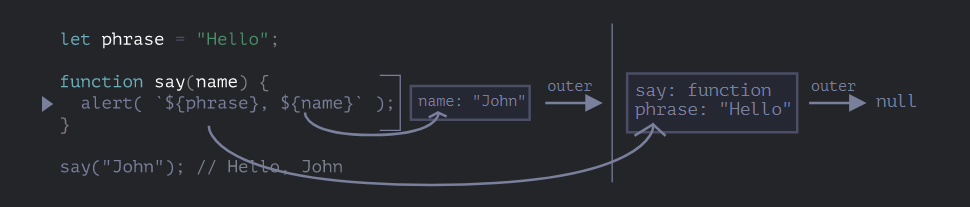
There are three forms of variables: *var*, *let*, and *const*.

Variables that use *var* are accessible by lower scopes, and when used within the scope of blocks they are raised to the scope immediately above.

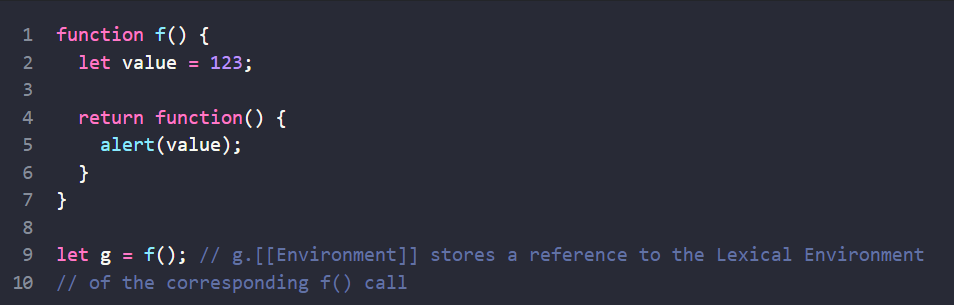
Variables that use *let* too are accessible by lower scopes but do *not suffer hoisting*, for example, if one variable use *let* inside one function, this variable will be available only inside this function.

Already when using the variable const have the same behavior that variable let, but the const value cannot be modified along the script processing.

Closure is a resource that works as a "memory" and remembers in which scope the function was created and for that it uses the Lexical Environment that is a specification object (value, properties, etc). When we call a function two lexical environments are created, an internal and an external one. The inner corresponds to the current execution and has a single property: the function argument. The external environment is the global Lexical Environment and has the variable and the function itself. When code wants to access a variable – the inner lexical environment is searched first, then the outer one, then the outermost one, and so on up to the global one.

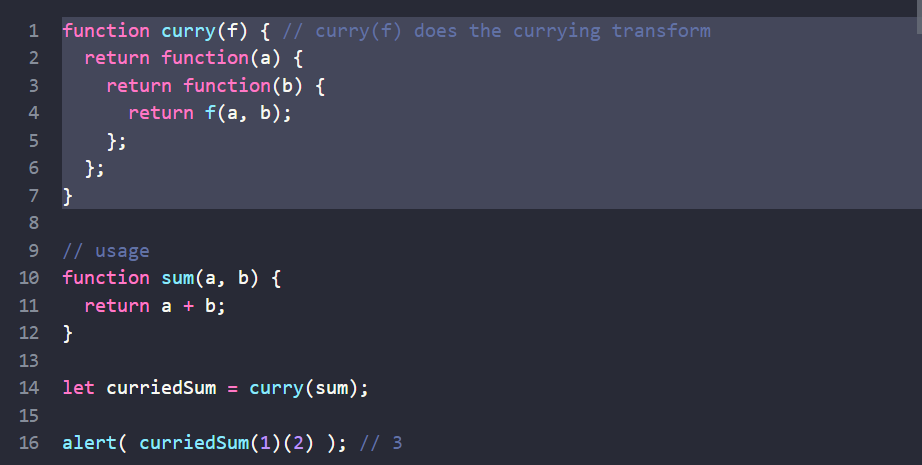


A nested function that is still accessible after the end of a function, it will have a property that references the lexical environment.[[Environment]]



# **Currying**

Currying is the way to transform a function that can have several parameters into just one function, that is, within the "main" function, another function will be returned with the desired parameter and so on.



[**CSS Animations**](https://javascript.info/css-animations)

It's the way to make animations only with CSS and be able to use Javascript to control and improve these animations. The transition resource is used to make the change in the indicated element and it has 4 properties:

* transition-property - determines the element where the animation will take place;
* transition-duration - deals with how long the animation will take;
* transition-delay - specifies the time (delay) to run the animation;
* transition-timing-function - determines how the animation will be during the run (changing speed, starts with a speed and then changes).

It can also use @keyframes to establish rules during animation execution.