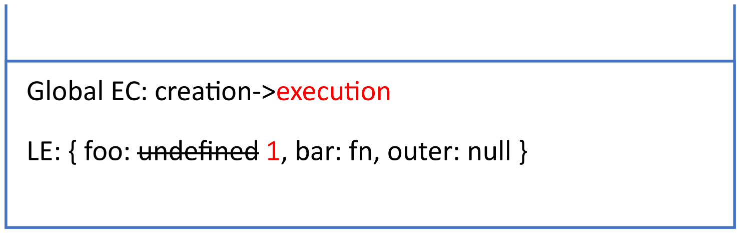
```
1: var foo = 1;
2: function bar() {
3:     if (!foo) {
4:         var foo = 10;
5:     }
6:     alert(foo)
7: }
8: bar()
```

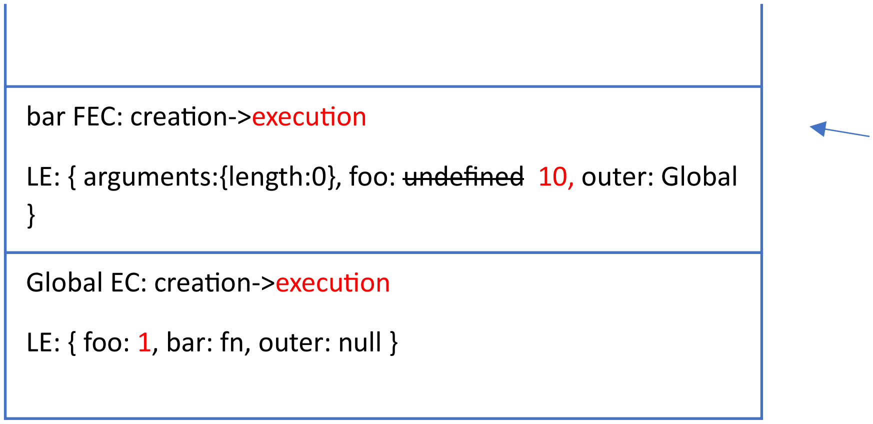
When a function is invoked, a function execution context(**FEC**) will be created for that function and pushed to the call stack. When the function finished/completed to execute, the function execution context will be popped off from the call stack.



Global EC: creation->execution

LE: { foo: ~~undefined~~ 1, bar: fn, outer: null }

Call Stack



bar FEC: creation->execution

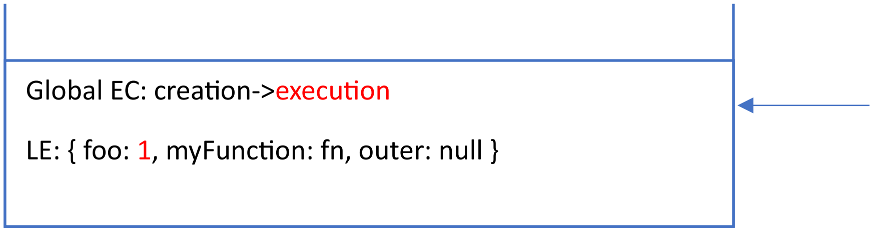
LE: { arguments:{length:0}, foo: ~~undefined~~ 10, outer: Global }

Global EC: creation->execution

LE: { foo: 1, bar: fn, outer: null }

Call Stack

bar() is invoked in line 8. FEC is created for the function and pushed to the call stack.



Global EC: creation->execution

LE: { foo: 1, myFunction: fn, outer: null }

Call Stack

bar FEC is popped off from the call stack after it is completed. The control is back to the Global EC.

Global EC creation: LE: { foo: undefined, bar: fn, outer: null }

Global EC execution: LE: { foo: 1, bar: fn, outer: null }

bar FEC creation: LE: { arguments:{length:0}, foo: undefined, outer: Global }

bar FEC execution: LE: { arguments:{length:0}, foo: 10, outer: Global }