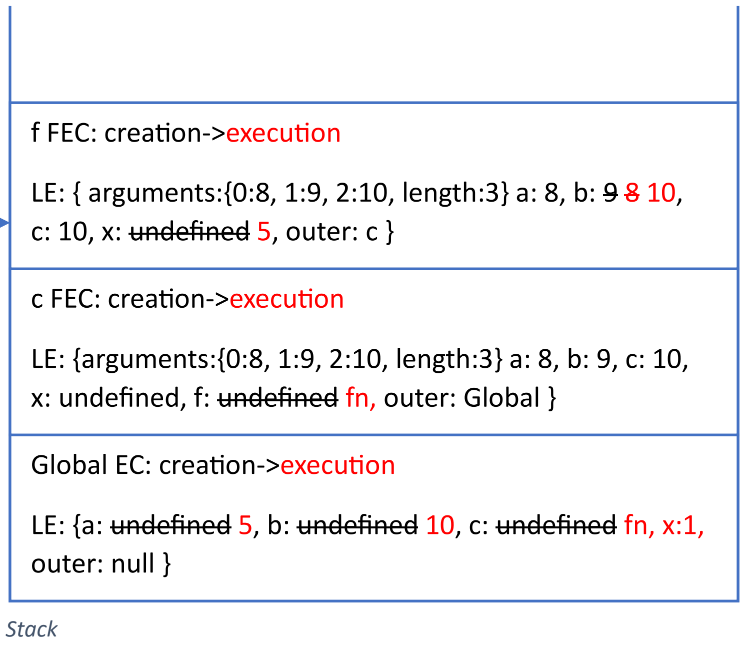
```
x = 1;
var a = 5;
var b = 10;
var c = function (a, b, c) {
    console.log(x);
    console.log(a);
    var f = function (a, b, c) {
        b = a;
        console.log(b);
        b = c;
        var x = 5;
    }
    f(a, b, c);
    console.log(b);
    var x = 10;
}
c(8, 9, 10);
console.log(b);
console.log(x);
```



f FEC: creation->execution

LE: { arguments:{0:8, 1:9, 2:10, length:3} a: 8, b: ~~9~~ ~~8~~ 10, c: 10, x: ~~undefined~~ 5, outer: c }

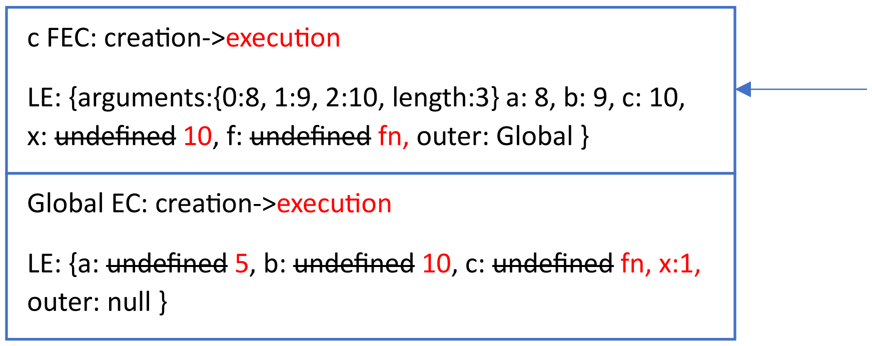
c FEC: creation->execution

LE: {arguments:{0:8, 1:9, 2:10, length:3} a: 8, b: 9, c: 10, x: undefined, f: ~~undefined~~ fn, outer: Global }

Global EC: creation->execution

LE: {a: ~~undefined~~ 5, b: ~~undefined~~ 10, c: ~~undefined~~ fn, x:1, outer: null }

Stack



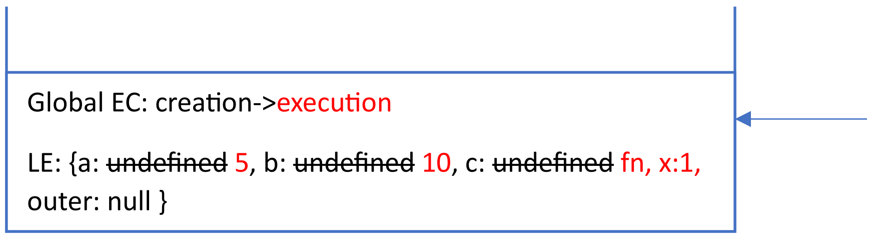
c FEC: creation->execution

LE: {arguments:{0:8, 1:9, 2:10, length:3} a: 8, b: 9, c: 10, x: ~~undefined~~ 10, f: ~~undefined~~ fn, outer: Global }

Global EC: creation->execution

LE: {a: ~~undefined~~ 5, b: ~~undefined~~ 10, c: ~~undefined~~ fn, x:1, outer: null }

f FEC is popped off from the call stack after it is completed. The control is back to the c function.



Global EC: creation->execution

LE: {a: ~~undefined~~ 5, b: ~~undefined~~ 10, c: ~~undefined~~ fn, x:1, outer: null }

c FEC is popped off from the call stack after it is completed. The control is back to the global scope.

Global EC creation: LE: {a: undefined, b: undefined, c: undefined, outer: null }

Global EC execution: LE: {a: 5, b: 10, c: fn, x=1, outer: null }

c FEC creation: LE: {arguments:{0:8, 1:9, 2:10, length:3}, a: 8, b: 9, c: 10, x: undefined, f: undefined, outer: Global }

c FEC execution: LE: {arguments:{0:8, 1:9, 2:10, length:3}, a: 8, b: 9, c: 10, x: undefined, f:fn, outer: Global }

f FEC creation: LE: { arguments:{0:8, 1:9, 2:10, length:3}, a: 8, b: 9, c: 10, x: undefined, outer: c }

f FEC execution: LE: { arguments:{0:8, 1:9, 2:10, length:3} a: 8, b: ~~8~~ 10, c: 10, x: 5, outer: c }