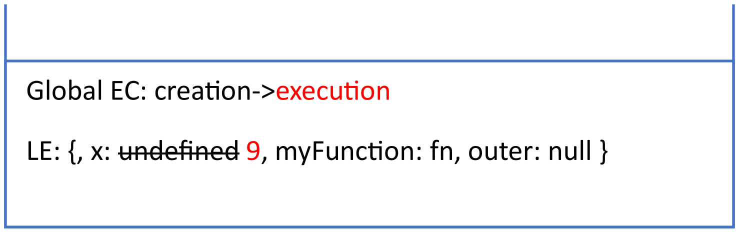
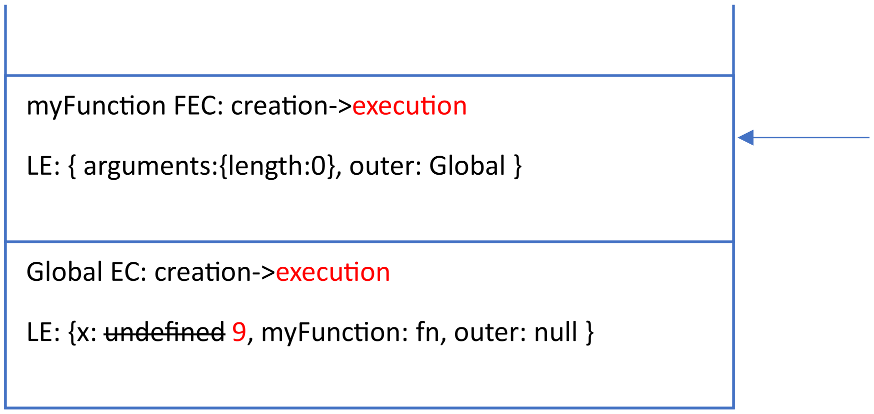
```
1: var x = 9;
2: function myFunction() {
3:     return x * x;
4: }
5: console.log(myFunction());
6: x = 5;
7: console.log(myFunction());
```

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.

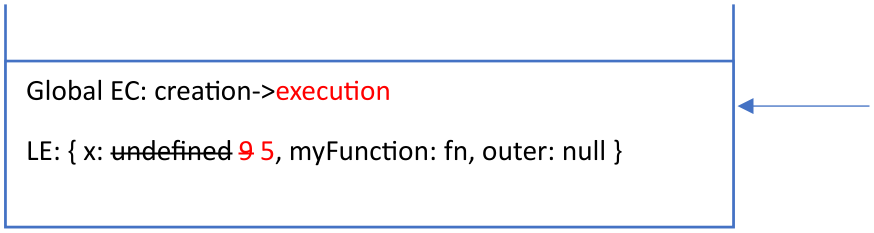

Global EC: creation->execution
LE: {, x: ~~undefined~~ 9, myFunction: fn, outer: null }

Call Stack


myFunction FEC: creation->execution
LE: { arguments:{length:0}, outer: Global }
Global EC: creation->execution
LE: {x: ~~undefined~~ 9, myFunction: fn, outer: null }

myFunction() is invoked in line 5. FEC is created for the function and pushed to the call stack.

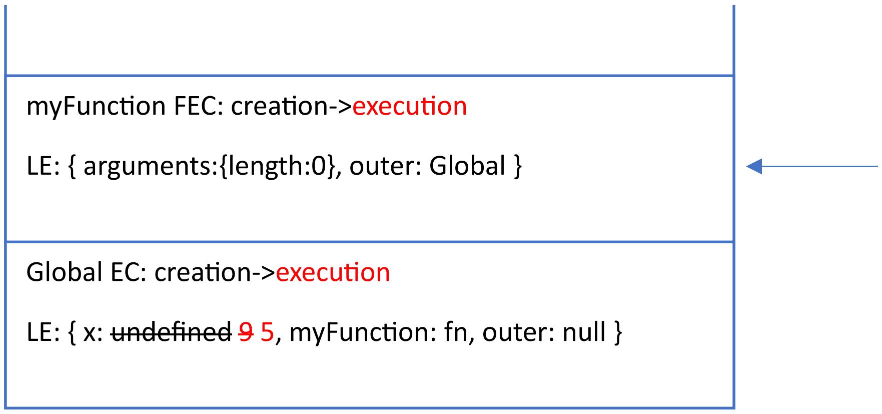
Call Stack


Global EC: creation->execution
LE: { x: ~~undefined~~ ~~9~~ 5, myFunction: fn, outer: null }

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

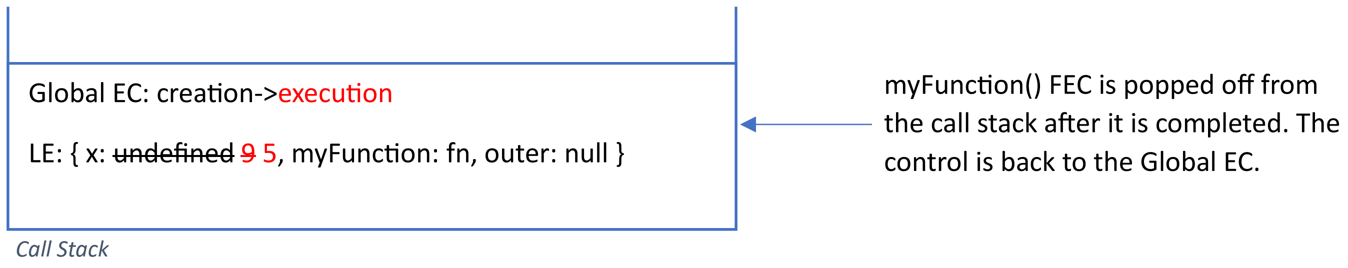
5 is assigned to x.

Call Stack


myFunction FEC: creation->execution
LE: { arguments:{length:0}, outer: Global }
Global EC: creation->execution
LE: { x: ~~undefined~~ ~~9~~ 5, myFunction: fn, outer: null }

myFunction() is invoked in line 7. FEC is created for the function and pushed to the call stack.

Call Stack



Global EC: creation->execution

LE: { x: ~~undefined~~ ~~9~~ 5, myFunction: fn, outer: null }

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

Call Stack

Global EC creation: LE: {, x: undefined, myFunction: fn, outer: null }

Global EC execution: LE: {, x: ~~9~~ 5, myFunction: fn, outer: null }

myFunction FEC creation: LE: { arguments:{length:0}, outer: Global }

myFunction FEC execution: LE: { arguments:{length:0}, outer: Global }

myFunction FEC creation: LE: { arguments:{length:0}, outer: Global }

myFunction FEC execution: LE: { arguments:{length:0}, outer: Global }