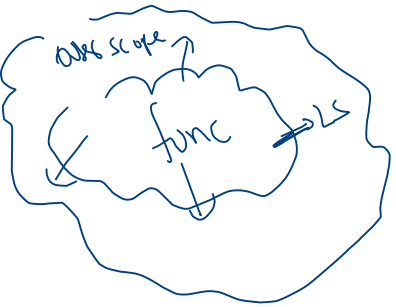
closures

↳ when a func^n remembers from where it came.

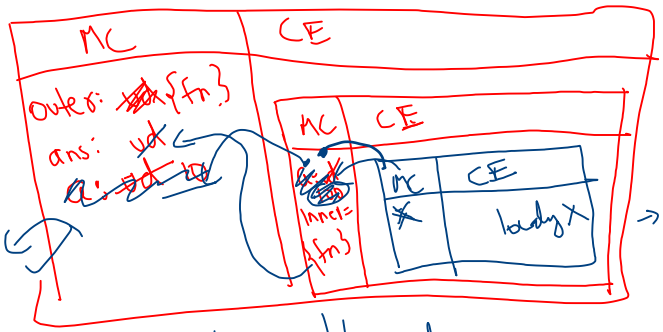
function along with it's local scope and it's lexical environment forms a closure.




```
// let a = 10;
function outer() {
    // let a = 100;
    function inner() {
        console.log(a)
    }

    return inner;
}

let ans = outer();
ans();
```

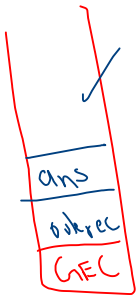


LST OLS GLS

inner

closure(outer)

LST Global



lexical scope chaining || lexical env

LE of inner = LS of inner + LE of outer → ③ Console

LE of outer = LS of outer + LE of global → ② ~~100~~ 10

LE of global = LS of global → ① Err: a is not defined

LE of inner = LS of inner + LS of outer + global

golden

Ch

function along with it's lexical scope forms a closure. In other words, a closure gives you access to outer function's scope in inner function