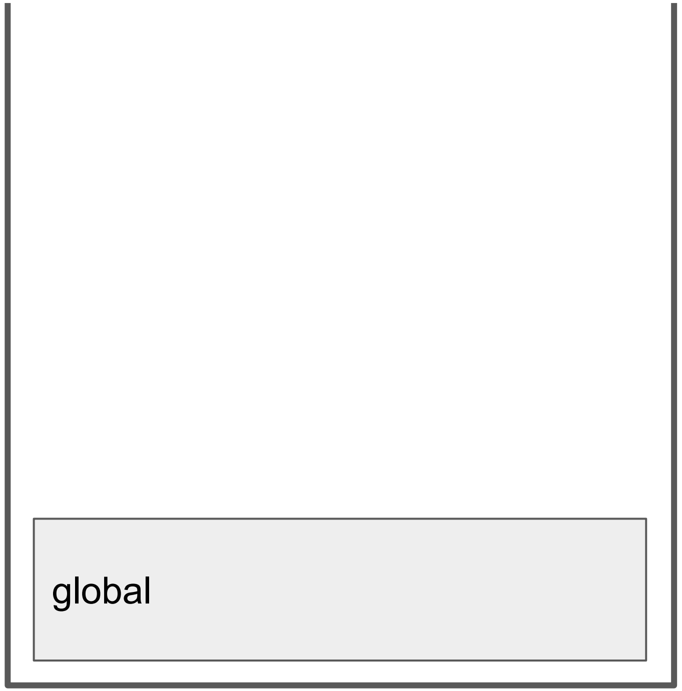
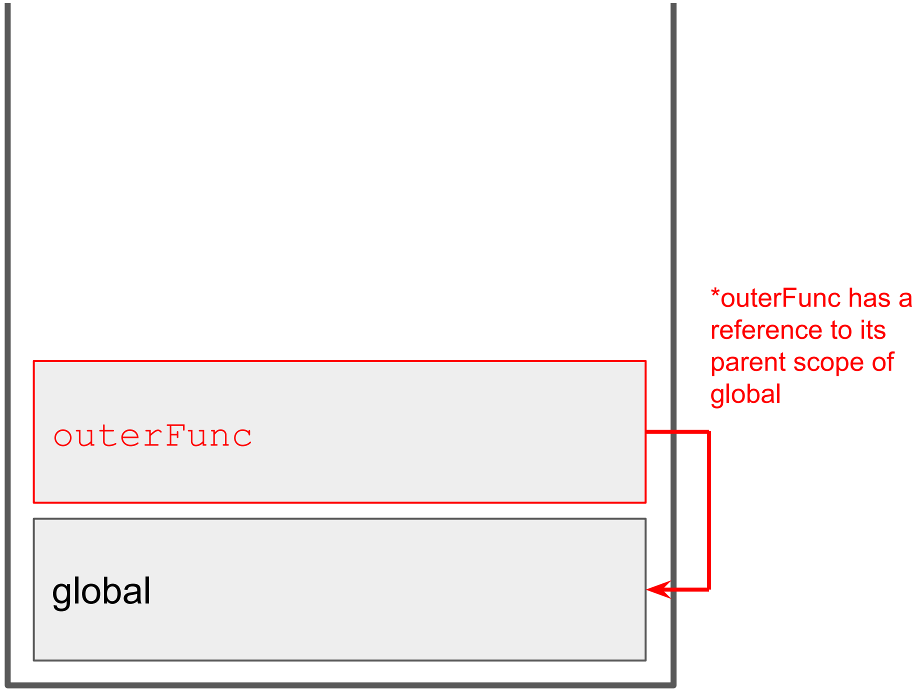
```
function outerFunc(){
    console.log("in the outer function");
    function innerFunc(){
        console.log("in the inner function")
    }
    anotherFunc();
    innerFunc();
}

function anotherFunc(){
    console.log("in the another function");
}

outerFunc();
```

Execution Stack - at the start of the program, the global scope is pushed onto the stack



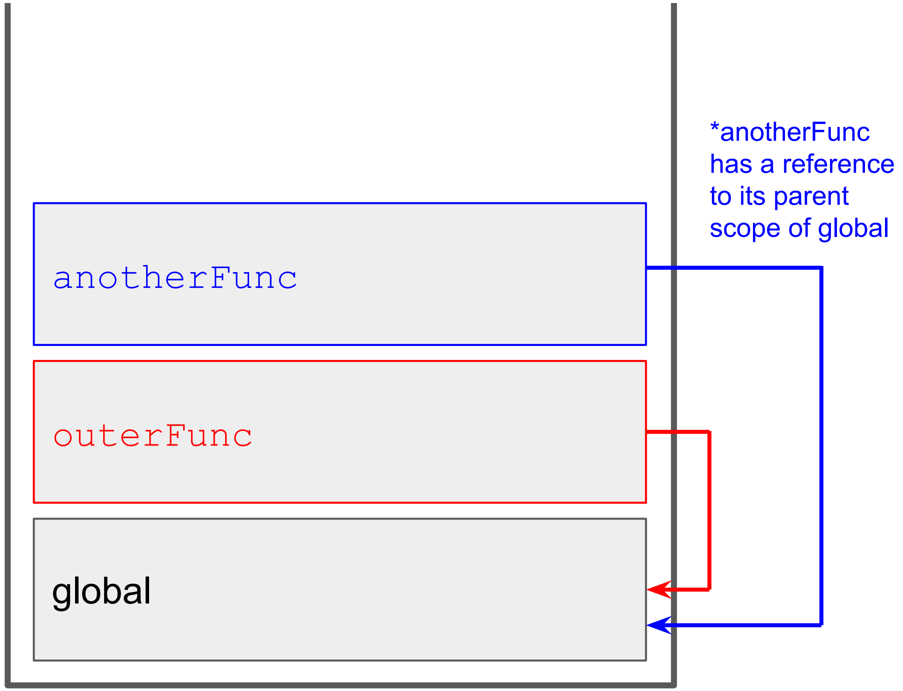
```
>
```

Execution Stack - outerFunc is invoked, so it is pushed onto the stack and executed



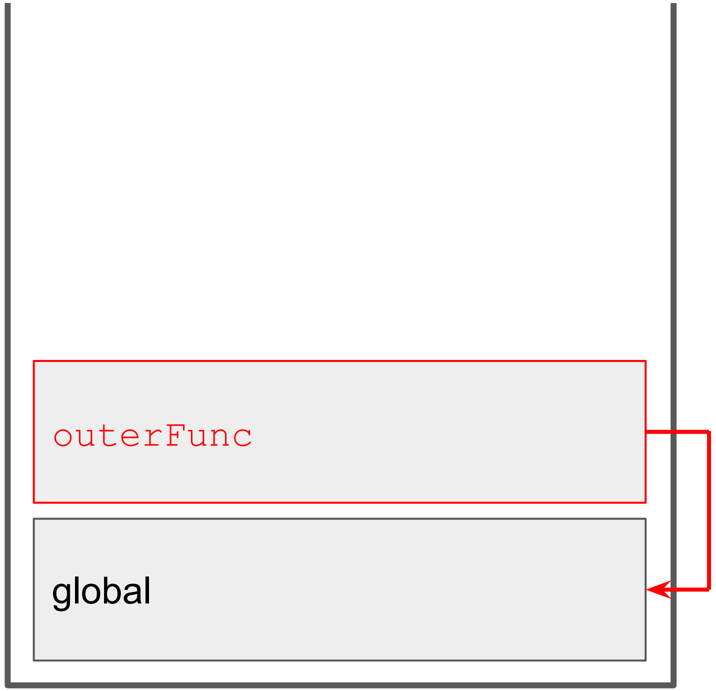
```
> in the outer function

>
```

Execution Stack - anotherFunc is invoked, so it is pushed onto the stack and executed



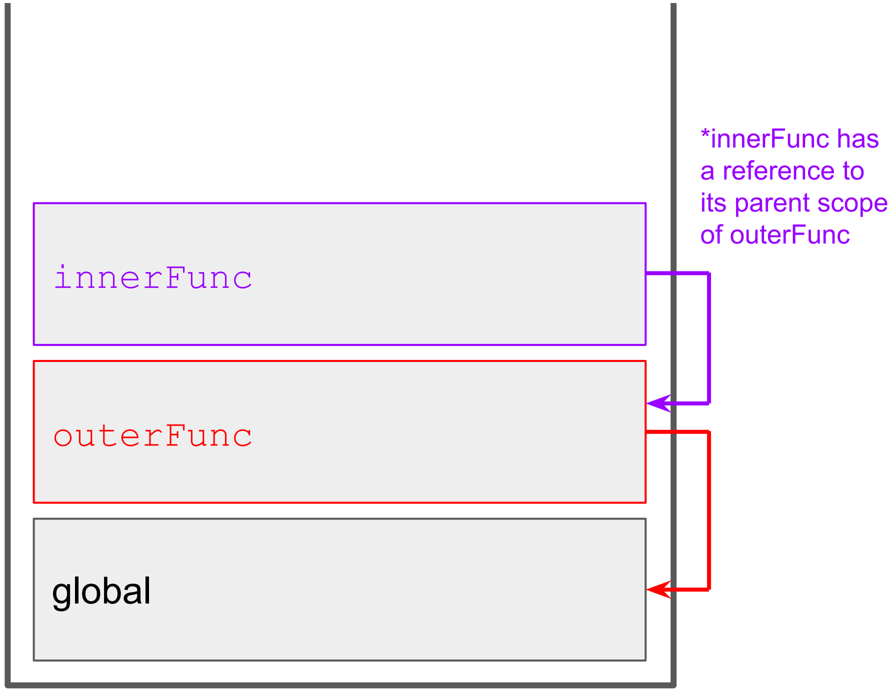
```
> in the outer function

> in the another function

>
```

Execution Stack - we reach the end of anotherFunc, so it is popped off the stack



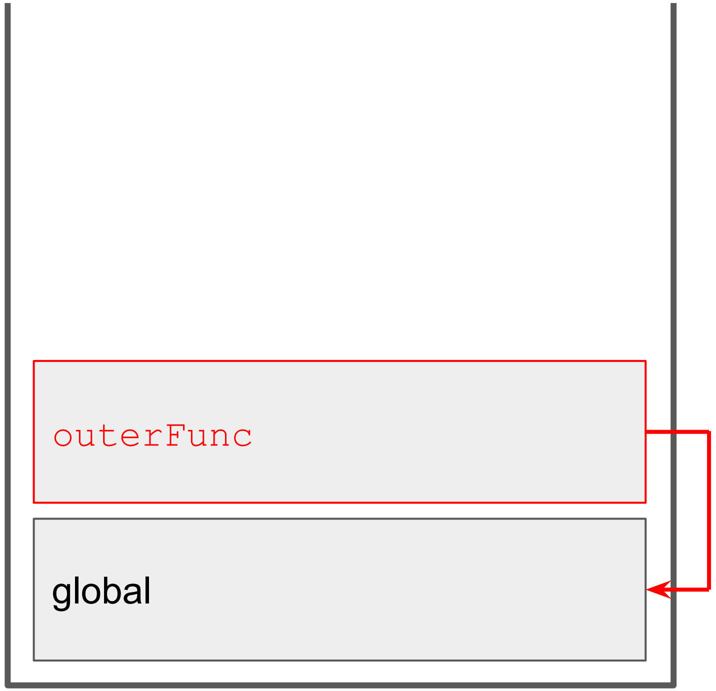
```
> in the outer function

> in the another function

>
```

Execution Stack - innerFunc is invoked, so it is pushed onto the stack and executed



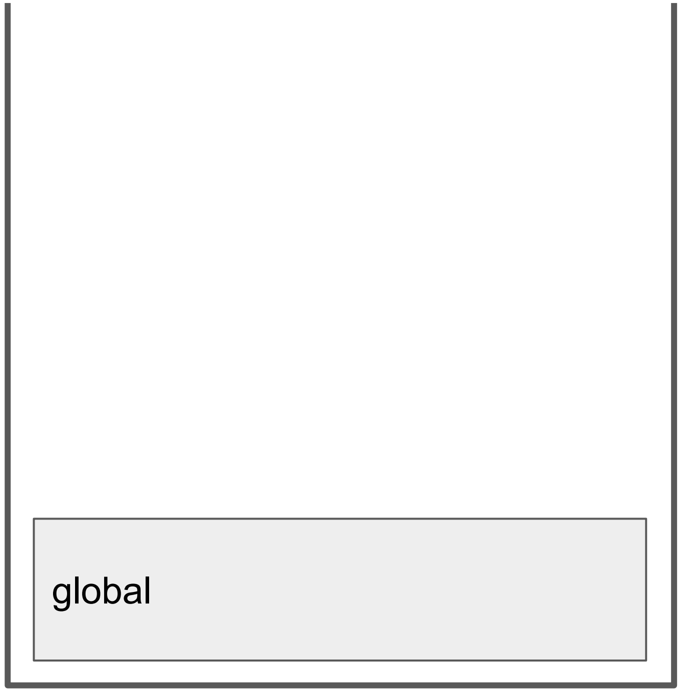
```
> in the outer function

> in the another function

> in the inner function

>
```

Execution Stack - we reach the end of innerFunc, so it is popped off the stack



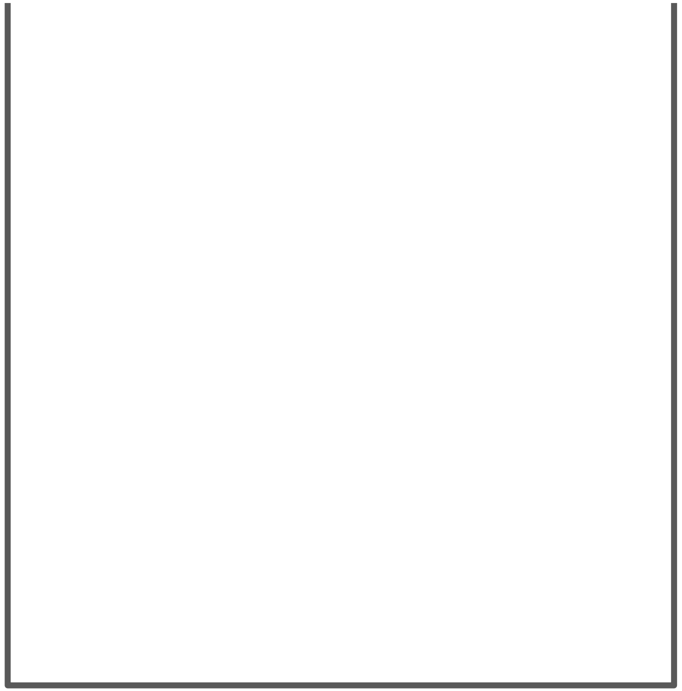
```
> in the outer function

> in the another function

> in the inner function

>
```

Execution Stack - we reach the end of outerFunc, so it is popped off the stack



```
> in the outer function

> in the another function

> in the inner function

>
```

Execution Stack - only when we reach the end of the entire program is global popped off



```
> in the outer function

> in the another function

> in the inner function

>
```