
CLOSURES

WHAT IS IT?

Process of storing an access to the latest state of a reference

```
1 function closured () {}
```

```
function closured () {
    debugger;
}
val = 121367512;
val2 = Math.random();
closured();
// show arguments, val and val2
val = 121367512;
val2 = Math.random();
closured();
```

LexicalEnvironment

Stack

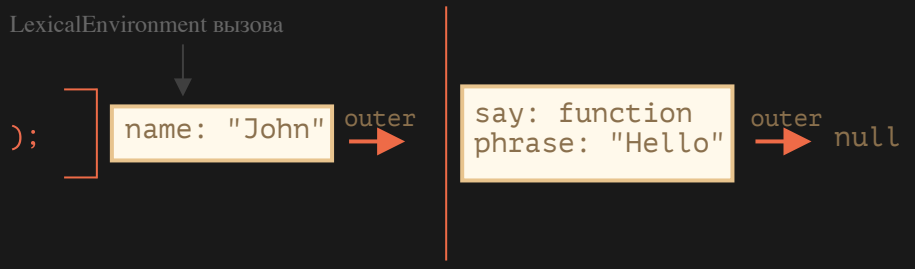
which is populated with variables of current scope and outer scope

till the Global object

```
let phrase = "Hello";

function say(name) {
  alert( `${phrase}, ${name}` );
}

say("John"); // Hello, John
```



source

When function instantiates **NOT** when they are run

```
const name = 'outside the function';
const getNameInModule = (function () {
    const name = 'within module function';
    function getName () { return name; };
    return getName;
})();

function getNameOutside () {
    return name;
};

getNameOutside(); // 'outside the function'
getNameInModule(); // 'within module function'
```

LATEST STATE

```
let name = 'outside';
function getNameOutside () {
    return name;
};
getNameOutside(); // 'outside';
name = 'latest';
getNameOutside(); // 'latest';
```

```
function counter (start = 0) {
    let count = start;
    return {
        increase: function (amount = 1) { count += amount }
        decrease: function (amount = 1) { count -= amount }
        getValue: function () { return count },
    };
};
const count = counter(10);
count.decrease();
count.decrease();
count.decrease();
count.getValue(); // 7
```

```
function counter (start = 0) {
    let count = start;
    return {
        increase: function (amount = 1) { count += amount }
        decrease: function (amount = 1) { count -= amount }
        getValue: function () { return count },
    };
};
const count = counter(10);
count.decrease();
count.decrease();
count.decrease();
count.getValue(); // 7
```

```
1 getValue: count
```

```
1 let count = Number;
2 getValue: count
```

```
let count = Number;
// Number is primitive
// primitive values are executed at once / copied
getValue: 0
```

PRIVATE METHODS

```
function counter (start = 0) {
    let count = start;
    return {
        increase: function (amount = 1) { count += amount }
        decrease: function (amount = 1) { count -= amount }
        getValue: function () { return count },
    };
};
const count = counter(10);
count.count // undefined
```

```
function counter (start = 0) {
    let count = start;
    function valueGetter () { return count }
    return {
        increase: function (amount = 1) { count += amount }
        decrease: function (amount = 1) { count -= amount }
        getValue: function returningValue () {
            return valueGetter() + start;
        }
    };
};
const count = counter(10);
count.increase(20);
count.increase(40);
count.getValue(); // 80
```

```
function counter (start = 0) {
    let count = start;
    function valueGetter () { return count }
    return {
        increase: function (amount = 1) { count += amount }
        decrease: function (amount = 1) { count -= amount }
        getValue: function returningValue () {
            return valueGetter() + start;
        }
    };
};
const count = counter(10);
count.increase(20);
count.increase(40);
count.getValue(); // 80
```

```
function counter (start = 0) {
    // ...
    function valueGetter () { return count }
    return {
        // ...
    };
};
const count = counter(10);
// ...
counter.valueGetter // undefined = private
```

COMMON MISTAKE IS LATEST STATE

```
1 const divs = document.querySelectorAll('div');
2 let index = 0;
3
4 for (; index < divs.length; index+=1) {
5     divs[index].onmouseover = function () { console.log(inde
6 }
```

```
1 const divs = document.querySelectorAll('div');
2
3 for (let index = 0; index < divs.length; index+=1) {
4     divs[index].onmouseover = function () { console.log(inde
5 }
```

```
1 const divs = document.querySelectorAll('div');
2
3 for (let index = 0; index < divs.length; index+=1) {
4     divs[index].onmouseover = function () { console.log(inde
5 }
```

```
const divs = document.querySelectorAll('div');
let index = 0;

for (; index < divs.length; index+=1) {
    (function (indexInternal) {
        // indexInternal is closured with the latest value
        divs[indexInternal].onmouseover = function () { cons
    })(index);
}
```

```
const divs = document.querySelectorAll('div');
let index = 0;

for (; index < divs.length; index+=1) {
    (function (indexInternal) {
        // indexInternal is closured with the latest value
        divs[indexInternal].onmouseover = function () { cons
    })(index);
}
```

```
const divs = document.querySelectorAll('div');
let index = 0;

for (; index < divs.length; index+=1) {
    divs[index].onmouseover = function () {
        console.log(arguments.callee.index)
    };
    divs[index].onmouseover.index = index;
}
```

```
const divs = document.querySelectorAll('div');
let index = 0;

for (; index < divs.length; index+=1) {
    divs[index].onmouseover = function () {
        'use strict';
        console.log(arguments.callee.index);
    };
    divs[index].onmouseover.index = index;
}
// Uncaught TypeError
```

```
const divs = document.querySelectorAll('div');
let index = 0;

for (; index < divs.length; index+=1) {
    divs[index].onmouseover = function named () {
        'use strict';
        console.log(named.index);
    };
    divs[index].onmouseover.index = index;
}
```

```
const divs = document.querySelectorAll('div');
let index = 0;

for (; index < divs.length; index+=1) {
    divs[index].onmouseover = function named () {
        'use strict';
        console.log(named.index);
    };
    divs[index].onmouseover.index = index;
}
```

```
const func = function (closuredValue) {
    return function closured () {
        // I have an access to `closuredValue`
        // and outer scope, which is `window`
    };
};
```