

A scenic coastal landscape featuring a rocky coastline, green hills, and a blue sky with white clouds. In the foreground, a metal railing gate stands closed, with a wooden post visible on the right. The word "CLOSURES" is overlaid in large, bold, white capital letters.

CLOSURES

WHAT IS IT?

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


```
1 function closured () {
2     debugger;
3 }
4 val = 121367512;
5 val2 = Math.random();
6 closured();
7 // show arguments, val and val2
8 val = 121367512;
9 val2 = Math.random();
10 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
```

LexicalEnvironment вызова



name: "John" outer

say: function
phrase: "Hello" outer null

source

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

```
1 const name = 'outside the function';
2 const getNameInModule = (function () {
3     const name = 'within module function';
4     function getName () { return name; };
5     return getName;
6 })();
7
8 function getNameOutside () {
9     return name;
10}
11
12 getNameOutside(); // 'outside the function'
13 getNameInModule(); // 'within module function'
```

LATEST STATE

```
1 let name = 'outside';
2 function getNameOutside () {
3     return name;
4 }
5 getNameOutside(); // 'outside';
6 name = 'latest';
7 getNameOutside(); // 'latest';
```

```
1 function counter (start = 0) {
2     let count = start;
3     return {
4         increase: function (amount = 1) { count += amount },
5         decrease: function (amount = 1) { count -= amount },
6         getValue: function () { return count },
7     };
8 };
9 const count = counter(10);
10 count.decrease();
11 count.decrease();
12 count.decrease();
13 count.getValue(); // 7
```

```
1 function counter (start = 0) {
2     let count = start;
3     return {
4         increase: function (amount = 1) { count += amount },
5         decrease: function (amount = 1) { count -= amount },
6         getValue: function () { return count },
7     };
8 };
9 const count = counter(10);
10 count.decrease();
11 count.decrease();
12 count.decrease();
13 count.getValue(); // 7
```



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

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

PRIVATE METHODS

```
1 function counter (start = 0) {
2     let count = start;
3     return {
4         increase: function (amount = 1) { count += amount }
5         decrease: function (amount = 1) { count -= amount }
6         getValue: function () { return count },
7     };
8 };
9 const count = counter(10);
10 count.count // undefined
```

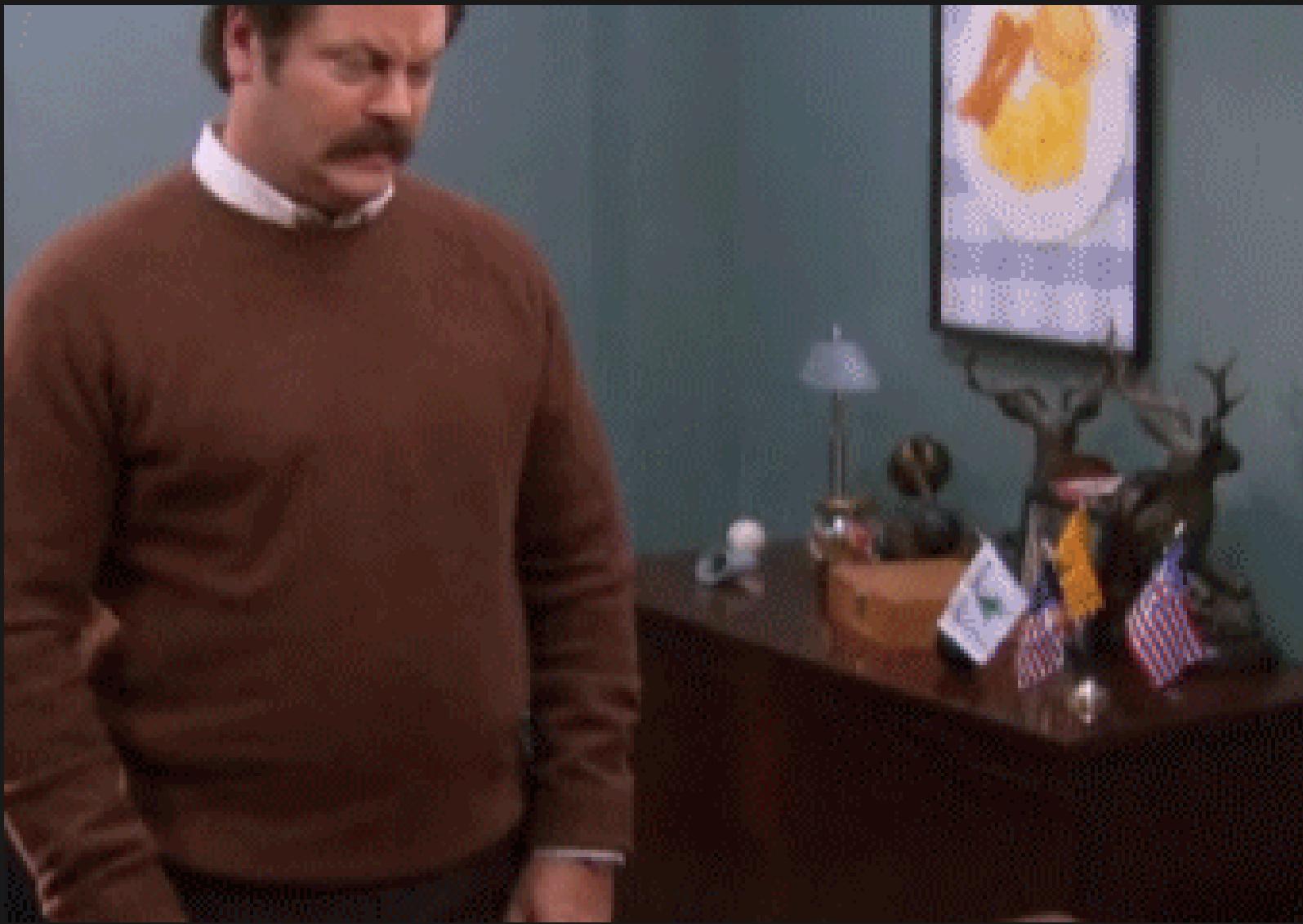
```
1 function counter (start = 0) {
2     let count = start;
3     function valueGetter () { return count }
4     return {
5         increase: function (amount = 1) { count += amount }
6         decrease: function (amount = 1) { count -= amount }
7         getValue: function returningValue () {
8             return valueGetter() + start;
9         }
10    };
11 };
12 const count = counter(10);
13 count.increase(20);
14 count.increase(40);
15 count.getValue(); // 80
```

```
1 function counter (start = 0) {
2     let count = start;
3     function valueGetter () { return count }
4     return {
5         increase: function (amount = 1) { count += amount }
6         decrease: function (amount = 1) { count -= amount }
7         getValue: function returningValue () {
8             return valueGetter() + start;
9         }
10    };
11 };
12 const count = counter(10);
13 count.increase(20);
14 count.increase(40);
15 count.getValue(); // 80
```

```
1 function counter (start = 0) {
2     // ...
3     function valueGetter () { return count }
4     return {
5         // ...
6     };
7 };
8 const count = counter(10);
9 // ...
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(index);
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(index);
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(index);
5 }
```

```
1 const divs = document.querySelectorAll('div');
2 let index = 0;
3
4 for (; index < divs.length; index+=1) {
5     (function (indexInternal) {
6         // indexInternal is closed with the latest value
7         divs[indexInternal].onmouseover = function () { cons
8     })(index);
9 }
```

```
1 const divs = document.querySelectorAll('div');
2 let index = 0;
3
4 for (; index < divs.length; index+=1) {
5     (function (indexInternal) {
6         // indexInternal is closed with the latest value
7         divs[indexInternal].onmouseover = function () { cons
8     })(index);
9 }
```

```
1 const divs = document.querySelectorAll('div');
2 let index = 0;
3
4 for (; index < divs.length; index+=1) {
5     divs[index].onmouseover = function () {
6         console.log(arguments.callee.index)
7     };
8     divs[index].onmouseover.index = index;
9 }
```

```
1 const divs = document.querySelectorAll('div');
2 let index = 0;
3
4 for (; index < divs.length; index+=1) {
5     divs[index].onmouseover = function () {
6         'use strict';
7         console.log(arguments.callee.index);
8     };
9     divs[index].onmouseover.index = index;
10 }
11 // Uncaught TypeError
```

```
1 const divs = document.querySelectorAll('div');
2 let index = 0;
3
4 for (; index < divs.length; index+=1) {
5     divs[index].onmouseover = function named () {
6         'use strict';
7         console.log(named.index);
8     };
9     divs[index].onmouseover.index = index;
10 }
```

```
1 const divs = document.querySelectorAll('div');
2 let index = 0;
3
4 for (; index < divs.length; index+=1) {
5     divs[index].onmouseover = function named () {
6         'use strict';
7         console.log(named.index);
8     };
9     divs[index].onmouseover.index = index;
10 }
```



```
1 const func = function (closedValue) {
2     return function closed () {
3         // I have an access to `closedValue`  

4         // and outer scope, which is `window`  

5     };
6 };
```