

**javascripti trikid**

ehk

**sulundid  
funktsioonid**

**prototüübid**

[github.com/egonelbre/jstricks](https://github.com/egonelbre/jstricks)  
[git.io/asDSQw](https://git.io/asDSQw)

Sometimes,  
the elegant implementation is a function.  
Not a method.  
Not a class.  
Not a framework.  
**Just a function.**

- John Carmack

heal lapsel mitu nime

lexical closure

lambda

anonymous function

first-class function

high-order function

delegate

blocks

...

# mitte ainult javascriptist...

<code>lisp</code>	<code>- (lambda (x) (x*x))</code>
<code>C++11</code>	<code>- [](int x){return x*x}</code>
<code>python</code>	<code>- lambda x: x*x</code>
<code>C#</code>	<code>- x =&gt; x*x</code>
<code>Obj.-C</code>	<code>- ^(int x){return x*x}</code>
<code>ruby</code>	<code>- lambda{ x  x*x}</code>
<code>haskell</code>	<code>- \x -&gt; x*x</code>
<code>java 8</code>	<code>- x =&gt; x*x</code>
<code>PHP</code>	<code>- create_function("\$x","return \$x*\$x")</code>
<code>...</code>	

# HOIATUS!

Järgnevaid asju nimetan "trikkideks" põhjusega.

With great power comes great responsibility.

Ehk enne live-süsteemis rakendamist veendu, et oled võimeline neid trikke debugima.

Võimalusel kasuta juba olemas olevaid asju, mis selliseid asju sinu eest ära teeb.

```
1 function map(arr, fun){
2     var r = [];
3     for(var i=0; i<arr.length; i+=1 )
4         r[i] = fun(arr[i]);
5     return r
6 }
7
8 map([1,2,3],
9     function(x){return x*x})
10 // [1,4,9]
```

```
13 power = function(p){
14     return function(x){
15         return Math.pow(x,p)
16     }
17 }
18 sqr = power(2)
19 sqr(3)
20
21 power(3)(2)
22 // 8
23 map([1,2,3], power(3))
```

**niih... alustame lihtsalt...  
loendamine....**

0

1

2

3

4

5

6

7

...

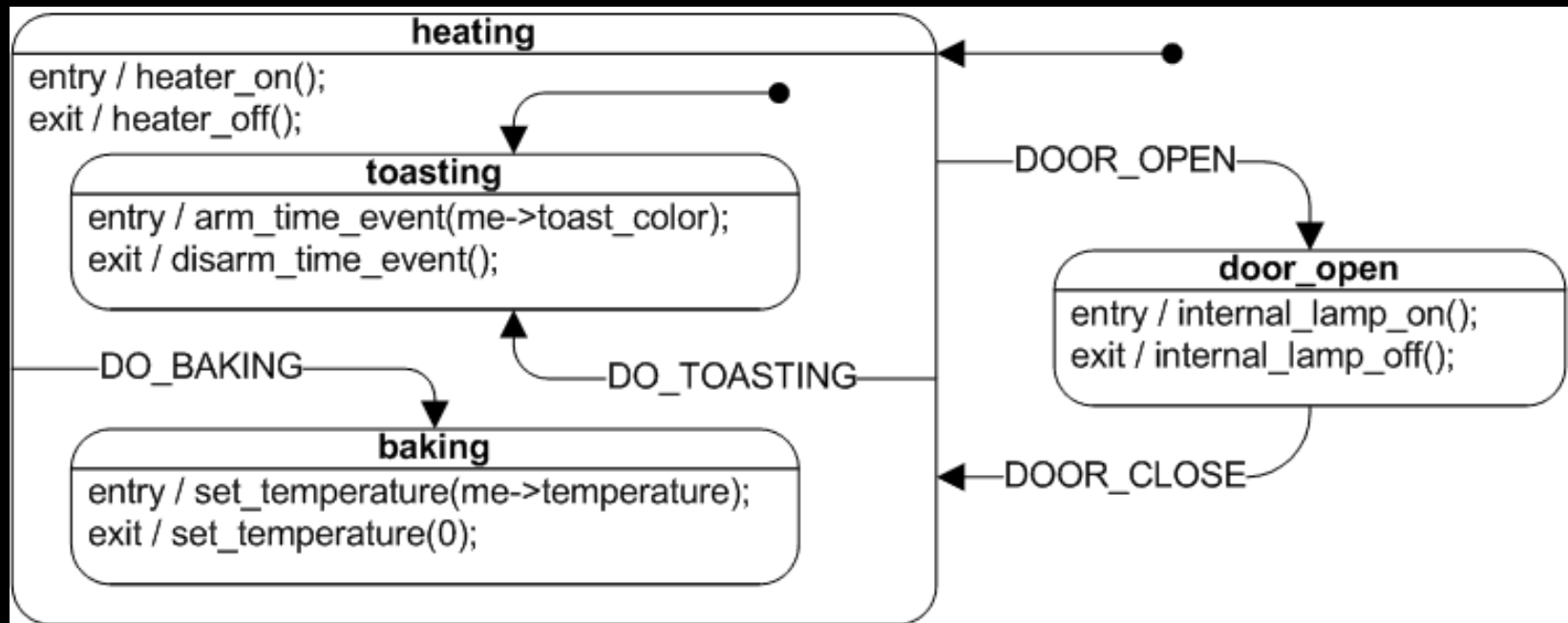


```
1 function Counter(start){
2     this.start = start - 1;
3 }
4
5 Counter.prototype.next = function(){
6     this.start += 1;
7     return this.start;
8 }
9
10 c = new Counter(0);
11 c.next() // 0
12 c.next() // 1
13 c.next() // 2
```

```
16 function NewIota(start){
17     start -= 1;
18     return function(){
19         start += 1;
20         return start;
21     }
22 }
23
24 iota = NewIota(0)
25 iota() // 0
26 iota() // 1
27 iota() // 2
```

```
29 // easy enumeration
30 enum = NewIota(0)
31 var itemTyp = {
32     Error : enum(),
33     Bool : enum(),
34     Complex : enum(),
35     EOF : enum()
36 }
```

# State Machine...



```
1  function Machine(){
2      this.cur = function(a){};
3      this.next = false;
4  }
5
6  Machine.prototype.update(){
7      if( this.next ){
8          this.cur("stop");
9          this.cur = this.next;
10         this.next = false;
11         this.cur("start");
12     }
13     this.cur("update");
14 }
15
```

```
35  
36 M = new Machine();  
37 M.next = M.def = function(s){  
38     if(mouse.down)  
39         this.next = this.line;  
40     mouse.render();  
41 };  
42  
43 M.line = function(){  
44     var start = {x:0,y:0}, last = {x:0,y:0};  
45     return function(s){  
46         if(s == "start")  
47             start = mouse.pos;  
48         last = mouse.pos;  
49         renderLine(start, last);  
50         if(!mouse.down)  
51             this.next = this.def;  
52         if(s == "stop")  
53             addLine(start, last);  
54     }  
55 }();  
56  
57 setInterval(M.update, 33);
```

# RPN kalkulaator

"3 4 + sin" == (3 4 +)sin

"3 4 5 \* -" == (3 (4 5 \*) -)

```
43 function ev(expr){
44     var stack = expr.split(/\s+/).reverse(),
45         args = [];
46     while (stack.length > 0){
47         token = stack.pop();
48         if( isNumber(token)) {
49             a = parseFloat(token);
50             args.push(a);
51         } else if( token == "+" ){
52             a = args.pop();
53             b = args.pop();
54             stack.push(a+b);
55         } ...
56         } else if (token == "sin" ){
57             a = args.pop();
58             stack.push(Math.sin(a));
59         } else {
60             throw "up";
61         }
62     }
63     return args;
64 }
65
66 result = ev("3 4 +");
67 log("result", result)
```



aga seda saab ju

**klassidega**

väga ilusasti teha

```
37  
38 function Add(){this.token="+"}  
39 Add.prototype.op=function(a,b){return a+b;}  
40 function Sub(){this.token="-"}  
41 Sub.prototype.op=function(a,b){return a-b;}  
42  
43 ops = [];  
44 ops.push(new Add())  
45 ops.push(new Sub())  
46  
47 findOp = function(ops, token){  
48     for(var i=0; i < ops.length; i+=1){  
49         if(ops[i].token == token)  
50             return ops[i];  
51     }  
52     return null;  
53 }  
54  
55 function ev(expr){  
56     var stack = expr.split(/\s+/).reverse(),  
57     args = [];  
58     while (stack.length > 0){  
59         token = stack.pop();  
60         if( isNumber(token)) {  
61             a = parseFloat(token);  
62             args.push(a);  
63         }  
64         else if( token == "(" ) {  
65             continue;  
66         }  
67         else if( token == ")" ) {  
68             op = findOp(ops, token);  
69             if( !op ) {  
70                 console.log("Invalid token: " + token);  
71                 return null;  
72             }  
73             a = args.pop();  
74             b = args.pop();  
75             res = op.op(a,b);  
76             args.push(res);  
77         }  
78     }  
79     return args[0];  
80 }
```

????

ILUSASTI

????

```
39 binOps = {
40     "+" : function(a,b){return a+b},
41     "-" : function(a,b){return a-b},
42     "*" : function(a,b){return a*b},
43     "/" : function(a,b){return a/b}
44 }
45
46 function ev(expr){
47     var stack = expr.split(/\s+/).reverse(),
48         args = [];
49     while (stack.length > 0){
50         token = stack.pop();
51         if( isNumber(token)) {
52             a = parseFloat(token);
53             args.push(a);
54         } else {
55             op = binOps[token];
56             if( typeof op == "undefined"){
57                 throw "up";
58             }
59             a = args.pop();
60             b = args.pop();
61             stack.push(op(a,b));
62         }
63     }
64     return args;
65 }
66
```

```
39 ops = {
40     "+" : [2,function(a){return a[0]+a[1]}],
41     "-" : [2,function(a){return a[0]-a[1]}],
42     "*" : [2,function(a){return a[0]*a[1]}],
43     "/" : [2,function(a){return a[0]/a[1]}],
44     "sin" : [1,function(a){return Math.sin(a[0])}],
45 }
46
47 function assert(v){ if(!v){throw "up"}}
48 function ev(expr){
49     var stack = expr.split(/\s+/).reverse(),
50         args = [];
51     while (stack.length > 0){
52         token = stack.pop();
53         if( isNumber(token)) {
54             a = parseFloat(token);
55             args.push(a);
56         } else {
57             op = ops[token];
58             assert(typeof op == "undefined");
59             a = args.splice(0,op[0]);
60             assert(a.length < op[0]);
61             stack.push(op[1](a));
62         }
63     }
64     return args;
65 }
```

hmmm....

```
function(a){return a[0]+a[1]}
```

liiga palju trükkimist....

teeks nii...

```
72  l("x[0]+x[1]")
73  ==
74  function(x){return x[0]+x[1]}
75
76
77  l("x*x")(5)
78  // 25
```

```
14  l = function(expr){
15      var body = "return (" + expr + ");";
16      return new Function("x", body);
17  }
18
19  sqr = l("x*x")
20  sqr(5)
21  // 25
22
```

(saab ka targema funktsiooni teha:

<http://osteele.com/sources/javascript/functional/> )



```
23 ops = {
24     "+" : [2, l("x[0]+x[1]")],
25     "-" : [2, l("x[0]-x[1]")],
26     "*" : [2, l("x[0]*x[1]")],
27     "/" : [2, l("x[0]/x[1]")],
28     "sin" : [1, l("Math.sin(x[0])")]
29 }
30
31 // or for extremely lazy
32 bin = function(o){return l("x[0]" + o + "x[1]")}
33 ops = {
34     "+" : [2, bin("+")],
35     "-" : [2, bin("-")],
36     "*" : [2, bin("*")],
37     "/" : [2, bin("/")],
38     "sin" : [1, l("Math.sin(x[0])")]
39 }
```

**aga igas keeles ei ole "eval"-i?**

... saab kombineerida funktsioone ...

```
45 obj = {
46     alpha : 4,
47     beta  : 10,
48     gamma : 123
49 }
50
51 function filterAlpha(arr, min, max){
52     r = [];
53     if((min !== NaN) && (max !== NaN)){
54         for(var i=0; i < arr.length; i += 1){
55             if( (arr[i].alpha >= min) && (arr[i].alpha <= max))
56                 r.push(arr[i])
57         }
58     } else if (min !== NaN){
59         for(var i=0; i < arr.length; i += 1){
60             if( (arr[i].alpha >= min))
61                 r.push(arr[i])
62         }
63     } else if (max !== NaN){
64         for(var i=0; i < arr.length; i += 1){
65             if( (arr[i].alpha <= max))
66                 r.push(arr[i])
67         }
68     }
69     ...
70 }
```

```
72 function filter(arr, fun){
73     r = [];
74     for(var i=0; i < arr.length; i += 1){
75         if ( fun(arr[i]) )
76             r.push(arr[i]);
77     }
78     return r;
79 }
80
81 function filterAlpha(arr, min, max){
82     if((min != NaN) && (max != NaN)){
83         return filter(arr, l("(x.alpha >= min) && (x.alpha <= max)"));
84     } else if (min != NaN){
85         return filter(arr, l("(x.alpha >= min)"));
86     } else if (max != NaN){
87         return filter(arr, l("(x.alpha <= max)"));
88     }
89 }
```

```
91 function filterer(value, min, max){
92     if((min != NaN) && (max != NaN)){
93         return function(x){
94             return (value(x) >= min) && (value(x) <= max);
95         };
96     } else if (min != NaN){
97         return function(x){ return (value(x) >= min); };
98     } else if (max != NaN){
99         return function(x){ return (value(x) <= max); };
100     } else {
101         return function(x){ return true; }
102     }
103 }
104
105 function filterAlpha(arr, min, max){
106     var fun = filterer(l("x.alpha"), min, max);
107     return filter(arr, fun);
108 }
109
110 function filterBeta(arr, min, max){
111     var fun = filterer(l("x.beta"), min, max);
112     return filter(arr, fun);
113 }
```

```

115
116 function filterer(valfun, min, max){
117     if((min != NaN) && (max != NaN)){
118         return function(x){
119             return (valfun(x) >= min) && (valfun(x) <= max);
120         };
121     } else if (min != NaN){
122         return function(x){ return (valfun(x) >= min); };
123     } else if (max != NaN){
124         return function(x){ return (valfun(x) <= max); };
125     } else {
126         return function(x){ return true; }
127     }
128 }
129
130 function makeFilter(value){
131     return function(arr, min, max){
132         var fun = filterer(l("x." + value), min, max);
133         return filter(arr, fun);
134     };
135 }
136
137 filterAlpha = makeFilter("alpha");
138 filterBeta  = makeFilter("beta");
139 filterGamma = makeFilter("gamma");

```

**sama põhimõttega...**

```
2  main = {
3      mouseAction: function(action, e){
4          if( action == "move" ) {
5              ...
6          }
7      },
8  };
9  input = {mouse:{down=false}};
10
11  mouseBinding = function(action){
12      return function(e){
13          if((action != "move") && (action != "wheel"))
14              input.mouse.down = action == "down";
15          main.mouseAction(action, e);
16          if(input.mouse.down)
17              modified();
18          e.preventDefault();
19      }
20  };
21
22  canvas.onmousemove=mouseBinding("move");
23  canvas.onmousedown=mouseBinding("down");
24  canvas.onmouseup=mouseBinding("up");
25  canvas.onmousewheel=mouseBinding("wheel");
26
```



# debugimise funktsioonid

```
4 myFunction = function(a,b,c){
5     b = a * c + b;
6     return a + b + c;
7 }
8
9 // ja debugima hakates
10
11 myFunction = function(a,b,c){
12     console.log("enter:", a,b,c);
13     b = a * c + b;
14     r = a + b + c;
15     console.log("exit:",a,b,c, "res:",r);
16     return r;
17 }
```

```
15 debug = function(fun, name){
16     return function(a,b,c){
17         console.log("enter:", name, a,b,c);
18         r = fun(a,b,c);
19         console.log("exit:", name, a,b,c, "res:", r);
20         return r;
21     }
22 }
23
24 myFunction = function(a,b,c){
25     b = a * c + b;
26     return a + b + c;
27 }
28
29 myFunction = debug(myFunction, "my");
30 myFunction(1,2,3);
31 // enter: my 1 2 3
32 // exit: my 1 2 3 res: 9
33 9
```

```
39 function toArray(args){
40     return Array.prototype.slice.call(args);
41 }
42
43 debug = function (fun, name){
44     return function(){
45         var args = toArray(arguments);
46         console.log("enter:", name, args);
47         var r = fun.apply(this, args);
48         console.log("exit:", name, args, 'res:', r);
49         return r;
50     }
51 };
52
53
54 myFunction = function(a,b,c,v){
55     b = a * c + b;
56     return a + b + c - v;
57 }
58
59 myFunction = debug(myFunction, "my");
60 log("result:", myFunction(1,2,3,4));
```

```
63
64 debugObject = function(obj, name){
65     for(var n in obj){
66         if(typeof obj[n] == "function"){
67             obj[n] = debug(obj[n], name + ":" + n)
68         }
69     }
70 }
71
72 obj = {
73     alpha : function(a,b){return a + b},
74     beta  : function(a,b,c){return a * b * c}
75 }
76
77 debugObject(obj, "obj");
78
79 obj.alpha(1,2);
80 obj.beta(1,2,3);
81
82 // enter: obj:alpha [1, 2]
83 // exit:  obj:alpha [1, 2] res: 3
84 // enter: obj:beta  [1, 2, 3]
85 // exit:  obj:beta  [1, 2, 3] res: 6
```

# lihtne logimine

```
37 // tavaliselt palju topelt
38 // trükkimist...
39 log.log("alpha", 1,2,3)
40 log.log("beta", 1,2,3)
41 log.log("zeta", 1)
42
43
44 // palju mõnusam oleks
45 log.alpha(1,2,3);
46 log.beta(1,2,3);
47 log.zeta(1,2,3);
```

```
16 function Log(){
17     this.log = function( name, data ){
18         console.log( name, ": ", data );
19     };
20
21     this.create = function(name){
22         var name = name,
23             logger = this;
24         if( typeof( this[ name ] ) !== "undefined" )
25             throw "Cannot use name : " + name + ". Already in use.";
26         this[name] = function(){ logger.log( name, arguments ) };
27     };
28 };
29
30 log = new Log();
31 log.create("render");
32 log.create("physics");
33 log.create("mouse");
34
35 log.render("alpha");
36 log.physics("1", 2, 3);
37 log.mouse("down", 100, 100);
```

```
42 function Log(){
43     this.enabled = {};
44
45     this.log = function( name, data ){
46         if( this.enabled[ name ] ){
47             console.log( name, ": ", data );
48         }
49     };
50
51     this.create = function(name, enabled){
52         var name = name,
53             logger = this;
54         if( typeof( this[ name ] ) !== "undefined" )
55             throw "Cannot use name : " + name + ". Already in use.";
56
57         logger[name] = function(){ logger.log( name, arguments ) };
58         logger.enabled[ name ] = enabled;
59         logger[name].enable = function(){logger.enable(name)};
60         logger[name].disable = function(){logger.disable(name)};
61     };
62
63     this.enable = function(name){ this.enabled[ name ] = true; };
64     this.disable = function(name){ this.enabled[ name ] = false; };
65 };
66
```



```
67  
68 log = new Log();  
69 log.create("physics", false);  
70  
71 log.physics.enable();  
72 log.physics("1", 2, 3);  
73 log.physics.disable();  
74 log.physics("1", 2, 3);  
75
```

# Lühidalt

Funktsioone manipuleerides saab koodi  
ilusamaks ning lühemaks.

Mine katseta omas keeles.

????