

master

...

fun-map / index.js / &lt;&gt; Jump to



edeflc fix assocIn/assocInM

History

1 contributor

212 lines (178 sloc) 4.44 KB

...

```
1 'use strict'; //jshint node:true
2 var has = {}.hasOwnProperty
3   , slice = [].slice
4
5 exports.get = get
6 function get(obj, key, fallback) {
7   return has.call(obj, key)
8     ? obj[key]
9     : fallback
10 }
11
12 exports.getIn = getIn
13 function getIn(obj, keys, fallback) { keys = keys.slice()
14   var key = keys.shift()
15   return keys.length
16     ? getIn(get(obj, key, {}), keys, fallback)
17     : get(obj, key, fallback)
18 }
19
20 exports.assoc = assoc
21 function assoc(obj, key, value) {
22   return arguments.length === 3
23     ? assocM(clone(obj), key, value)
24     : assoc.apply(null, arguments)
25 }
26
27 assoc.apply = function(_, args) {
28   var obj = args[0]
29   , ret = clone(obj)
30   for (var i = 1, len = args.length; i < len; i += 2)
31     assocM(ret, args[i], args[i + 1])
32
33   if (i !== len)
34     throw new Error('missing key')
35
36   return ret
37 }
38
39 exports.assocIn = assocIn
40 function assocIn(obj, keys, value) { return assocIn_(obj, keys.slice(), value) }
41 function assocIn_(obj, keys, value) {
42   var key = keys.shift()
43   return keys.length
44     ? assoc(obj, key, assocIn_(obj[key] || {}, keys, value))
45     : assoc(obj, key, value)
46 }
47
48 exports.dissoc = dissoc
49 function dissoc(obj, key) {
50   if (arguments.length === 1)
51     return obj
52   else if (arguments.length <= 4) {
53     var ret = {}
54     , keyA = key ? (key + '') : ''
55     , keyB = keyB ? (keyB + '') : ''
56     , keyC = keyC ? (keyC + '') : ''
57     for (var curKey in obj) if (has.call(obj, curKey))
58       if (curKey !== keyA && curKey !== keyB && curKey !== keyC)
59         ret[curKey] = obj[curKey]
60     return ret
61   }
62   else
63     return dissoc.apply(null, arguments)
64 }
65
66 dissoc.apply = function(_, args) {
67   var obj = clone(args[0])
68   for (var i = 1, len = args.length; i < len; i++)
69     dissocM(obj, args[i])
70   return obj
71 }
72
73 exports.merge = merge
74 function merge(obj, src) {
75   return reduce(_mergeM, {}, slice.call(arguments))
76 }
77
78 exports.zipmap = zipmap
```

```

79 function zipmap(keys, vals) {
80   var ret = {}
81   for (var i = 0, len = keys.length; i < len; i++)
82     ret[keys[i]] = vals[i]
83   return ret
84 }
85
86 exports.selectKeys = selectKeys
87 function selectKeys(obj, keys) {
88   var ret = {}
89   for (var i = 0, len = keys.length; i < len; i++) {
90     var key = keys[i]
91     if (has.call(obj, key))
92       ret[key] = obj[key]
93   }
94   return ret
95 }
96
97 var keys = exports.keys = Object.keys || function keys(obj) {
98   var ret = []
99   for (var key in obj) if (has.call(obj, key))
100     ret.push(key)
101   return ret
102 }
103
104 exports.vals = vals
105 function vals(obj) {
106   var ret = []
107   for (var key in obj) if (has.call(obj, key))
108     ret.push(obj[key])
109   return ret
110 }
111
112 exports.keyvals = keyvals
113 function keyvals(obj) {
114   var ret = []
115   for (var key in obj) if (has.call(obj, key))
116     ret.push([key, obj[key]])
117   return ret
118 }
119
120 exports.hashmap = hashmap
121 function hashmap() { return hashmap.apply(null, arguments) }
122
123 hashmap.apply = function(., keyvals) {
124   var ret = {}
125   for (var i = 0, len = keyvals.length; i < len; i += 2)
126     assocM(ret, keyvals[i], keyvals[i + 1])
127
128   if (i !== len)
129     throw new Error('missing key')
130
131   return ret
132 }
133
134 // mutations ahead!
135
136 exports.assocInM = assocInM
137 function assocInM(obj, keys, value) {
138   var ret = obj
139   , key
140
141   for (var i = 0, len = keys.length; i < (len - 1); i++) {
142     key = keys[i]
143     obj = obj[key] || (obj[key] = {})
144   }
145
146   key = keys[i]
147   obj[key] = value
148
149   return ret
150 }
151
152 exports.assocM = assocM
153 function assocM(obj, key, value) {
154   if (arguments.length === 3) {
155     obj[key] = value
156     return obj
157   }
158   return assocM.apply(null, arguments)
159 }
160
161 assocM.apply = function(., args) {
162   var obj = args[0]
163   for (var i = 1, len = args.length; i < len; i += 2)
164     assocM(obj, args[i], args[i + 1])
165
166   if (i !== len)
167     throw new Error('missing key')
168
169   return obj
170 }
171
172 exports.dissocM = dissocM
173 function dissocM(obj, key) {
174   if (arguments.length > 2)
175     return dissocM.apply(null, arguments)
176   delete obj[key]

```

```
177     return obj
178 }
179
180 dissocM.apply = function(_, args) {
181     var obj = args[0]
182     for (var i = 1, len = args.length; i < len; i++)
183         delete obj[args[i]]
184     return obj
185 }
186
187 exports.mergeM = mergeM
188 function mergeM(obj, src) {
189     return arguments.length === 2
190         ? _mergeM(obj, src)
191         : reduce(_mergeM, obj, slice.call(arguments, 1))
192 }
193
194 function _mergeM(obj, src) {
195     for (var key in src) if (has.call(src, key))
196         obj[key] = src[key]
197     return obj
198 }
199
200 function reduce(fn, initial, values) {
201     var acc = initial
202     for (var i = 0, len = values.length; i < len; i++)
203         acc = fn(acc, values[i])
204     return acc
205 }
206
207 function clone(obj) {
208     var ret = {}
209     for (var key in obj) if (has.call(obj, key))
210         ret[key] = obj[key]
211     return ret
212 }
```