dominictarr / **libnested** Public

Code · Issues · Pull requests · Actions · Projects · Security · Insights

d028a1b0f2

libnested / **index.js** / Jump to

dominictarr create arrays if the next key is an integer >= 0 · History

2 contributors

100 lines (90 sloc) | 2.48 KB

```js
function isObject (o, allowArray) {
  return o && 'object' === typeof o && (allowArray || !Array.isArray(o))
}

function isBasic (b) {
  return 'string' === typeof b || 'number' === typeof b
}

function get (obj, path, dft) {
  if(!isObject(obj, true)) return dft
  if(isBasic(path)) return obj[path]
  for(var i = 0; i < path.length; i++) {
    if(null == (obj = obj[path[i]])) return dft
  }
  return obj
}

function isNonNegativeInteger (i) {
  return Number.isInteger(i) && i >= 0
}

function set (obj, path, value) {
  if(!obj) throw new Error('libnested.set: first arg must be an object')
  if(isBasic(path)) return obj[path] = value
  for(var i = 0; i < path.length; i++)
    if(i === path.length - 1)
      obj[path[i]] = value
    else if(null == obj[path[i]])
      obj = (obj[path[i]] = isNonNegativeInteger(path[i+1]) ? [] : {})
    else
      obj = obj[path[i]]
  return value
}

function each (obj, iter, includeArrays, path) {
  path = path || []
  //handle array separately, so that arrays can have integer keys
  if(Array.isArray(obj)) {
    if(!includeArrays) return false
    for(var k = 0; k < obj.length; k++) {
      //loop content is duplicated, so that return works
      var v = obj[k]
      if(isObject(v, includeArrays)) {
        if(false === each(v, iter, includeArrays, path.concat(k)))
          return false
      } else {
        if(false === iter(v, path.concat(k))) return false
      }
    }
  }
  else {
    for(var k in obj) {
      //loop content is duplicated, so that return works
      var v = obj[k]
      if(isObject(v, includeArrays)) {
        if(false === each(v, iter, includeArrays, path.concat(k)))
          return false
      } else {
        if(false === iter(v, path.concat(k))) return false
      }
    }
  }
  return true
}

function map (obj, iter, out, includeArrays) {
  var out = out || Array.isArray(obj) ? [] : {}
  each(obj, function (val, path) {
    set(out, path, iter(val, path))
  }, includeArrays)
  return out
}

function paths (obj, incluedArrays) {
  var out = []
  each(obj, function (_, path) {
    out.push(path)
  }, incluedArrays)
  return out
}

function id (e) { return e }

//note, cyclic objects are not supported.
//will cause an stack overflow.
function clone (obj) {
  if(!isObject(obj, true)) return obj
  var _obj
  _obj = Array.isArray(obj) ? [] : {}
  for(var k in obj) _obj[k] = clone(obj[k])
  return _obj
}

exports.get = get
exports.set = set
exports.each = each
exports.map = map
exports.paths = paths
exports.clone = clone
exports.copy = clone
```