Home

Fast Map extension for Node.js & The Browser

LightMap

LightMap is an extension of Map that adds iterative extensions to Map like `map`, `filter`, `reduce`, `sort`, etc.

NPM

Installation

```
npm i @mi-sec/lightmap
```

Usage

Basic usage:

```
const LightMap = require( '@mi-sec/lightmap' );

const x = new LightMap( [ [ 'a', 'hello' ], [ 'b', 'world'
// LightMap { 'a' => 'hello', 'b' => 'world' }
```

Map usage:

For standard `Map` usage please refer to MDN Web Docs

`.filter`:

Filter LightMap based on keys and values based in. Comparisons can be made on the key and/or value.

```
const x = new LightMap( [ [ 'a', 'hello' ], [ 'b', 'world'

x.filter( ( val, key ) => {
        return key === 'a';
} );
// LightMap { 'a' => 'hello' }
```

`.map`:

Map LightMap with new key and/or value. Return the new item in a "tuple" form matching the Map paradigm (`[ x, y ]`).

```
const x = new LightMap( [ [ 'a', 10 ], [ 'b', 20 ] ] );

x.map( ( val, key ) => {
        return [ key, val + 10 ];
} );
// LightMap { 'a' => 20, 'b' => 30 }
```

`.reduce`:

Reduce LightMap with new value. Must return the carriage value just like Array.reduce.

```
const x = new LightMap( [ [ 'a', 'hello' ], [ 'b', 'world'

x.reduce( ( r, [ key, val ] ) => {
        r += val + ' ';
        return r;
}, '' );
// hello world
```

`.sortKeys`:

Map LightMap with sorted key-value pairs.

```
const x = new LightMap( [ [ 'b', 'world' ], [ 'a', 'hello'

x.sortKeys()
// LightMap { 'a' => 'hello', 'b' => 'world' }
```

`.sortValues`:

Map LightMap with sorted key-value pairs sorted by value.

```
const x = new LightMap( [ [ 'a', 10 ], [ 'b', 5 ], [ 'c', 1

x.sortValues( ( a, b ) => a >= b )
// LightMap { 'c' => 1, 'b' => 5, 'a' => 10 }

x.sortValues( ( a, b ) => a <= b )
// LightMap { 'a' => 10, 'b' => 5, 'c' => 1 }
```

`.mapToArray`:

maps a LightMap object to an array of arrays in the Map Pattern (re-constructable pattern)

```
const x = new LightMap( [ [ 'a', 0 ], [ 'b', 1 ] ] );

x.mapToArray();
// [ [ 'a', 0 ], [ 'b', 1 ] ]
```

`.toJSON`:

Native class override - returns `.mapToArray` method

```
const x = new LightMap( [ [ 'a', 0 ], [ 'b', 1 ] ] );

x.toJSON();
// [ [ 'a', 0 ], [ 'b', 1 ] ]
```

`.toString`:

Native class override - returns JSON stringified `.mapToArray`

```
const x = new LightMap( [ [ 'a', 0 ], [ 'b', 1 ] ] );

x.toString();
// [["a",0],["b",1]]
```

`.indexOf`:

returns the first index at which a given element can be found in the array, or -1 if it is not present

```
const x = new LightMap( [ [ 'a', 0 ], [ 'b', 1 ] ] );

x.indexOf( 'b' );
// 1
```

`[ Symbol.replace ]`:

symbol specifies the method that replaces matched substrings of a string

```
const x = new LightMap( [ [ '{{ a }}', 'hello' ], [ '{{ b 

'{{ a }} {{ b }}'.replace( x );
// hello world
```

`[ Symbol.toPrimitive( string ) ]`:

symbol that specifies a function valued property that is called to convert an object to a primitive value

```
const x = new LightMap( [ [ 'a', 0 ], [ 'b', 1 ] ] );

'' + x;
// [["a",0],["b",1]]
```

`[ Symbol.toPrimitive( number ) ]:`

symbol that specifies a function valued property that is called to convert an object to a primitive value

```
const x = new LightMap( [ [ 'a', 0 ], [ 'b', 1 ] ] );

+x;
// 2 (size of the Map)
```
