Lodash

JavaScript utility library delivering modularity, performance & extras

What is Lodash and why to use it!

- A modern JavaScript utility library delivering modularity, performance & extras

- Developed in 2012 by John-David Dalton under MIT license

- Lodash is a JavaScript library that helps programmers write more concise and easier to maintain JavaScript code

- It cover’s lots of areas like arrays, dates, functions, math, collections, objects, strings, etc.

Installing and using

- in a browser

```
<script src="lodash.js"></script>
```

- using npm

```
$ npm i -g npm
$ npm i --save lodash
```

- in code

```
// Load the full build.
var _ = require('lodash');

// Load method categories.
var array = require('lodash/array');
var object = require('lodash/fp/object');
```

Arrays

- `_.chunk(array, [size=1])`

- Creates an array of elements split into groups the length of size. If array can't be split evenly, the final chunk will be the remaining elements.

```
_.chunk(['a', 'b', 'c', 'd'], 2);
// => [['a', 'b'], ['c', 'd']]

_.chunk(['a', 'b', 'c', 'd'], 3);
// => [['a', 'b', 'c'], ['d']]
```

- `_.concat(array, [values])`

- Creates a new array concatenating array with any additional arrays and/or values.

```
var array = [1];
var other = _.concat(array, 2, [3], [[4]]);

console.log(other);
// => [1, 2, 3, [4]]

console.log(array);
// => [1]
```

- `_.pull(array, [values])`

- Removes all given values from array using SameValueZero for equality comparisons.

```
var array = ['a', 'b', 'c', 'a', 'b', 'c'];

_.pull(array, 'a', 'c');
console.log(array);
// => ['b', 'b']
```

- `_.union([arrays])`

- Creates an array of unique values, in order, from all given arrays using SameValueZero for equality comparisons.

```
_.union([2], [1, 2]);
// => [2, 1]
```

- `_.uniq(array)`

- Creates a duplicate-free version of an array, using SameValueZero for equality comparisons, in which only the first occurrence of each element is kept. The order of result values is determined by the order they occur in the array.

```
_.union([2], [1, 2]);
// => [2, 1]
```

Collection

- _.countBy(collection, [`iteratee=_.identity`])

- Creates an object composed of keys generated from the results of running each element of collection thru iteratee. The corresponding value of each key is the number of times the key was returned by iteratee.

```
_.countBy([6.1, 4.2, 6.3], Math.floor);
// => { '4': 1, '6': 2 }

// The `_.property` iteratee shorthand.
_.countBy(['one', 'two', 'three'], 'length');
// => { '3': 2, '5': 1 }
```

- `_.groupBy(collection, [iteratee=_.identity])`

- Creates an object composed of keys generated from the results of running each element of collection thru iteratee. The order of grouped values is determined by the order they occur in collection.

```
_.groupBy([6.1, 4.2, 6.3], Math.floor);
// => { '4': [4.2], '6': [6.1, 6.3] }

// The `_.property` iteratee shorthand.
_.groupBy(['one', 'two', 'three'], 'length');
// => { '3': ['one', 'two'], '5': ['three'] }
```

- `_.includes(collection, value, [fromIndex=0])`

- Checks if value is in collection.

```
_.includes([1, 2, 3], 1);
// => true

_.includes([1, 2, 3], 1, 2);
// => false

_.includes({ 'a': 1, 'b': 2 }, 1);
// => true

_.includes('abcd', 'bc');
// => true
```

Object

- `_.assign(object, [sources])`

- Assigns own enumerable string keyed properties of source objects to the destination object. Source objects are applied from left to right. Subsequent sources overwrite property assignments of previous sources.

```
function Foo() {
  this.a = 1;
}

function Bar() {
  this.c = 3;
}

Foo.prototype.b = 2;
Bar.prototype.d = 4;

_.assign({ 'a': 0 }, new Foo, new Bar);
// => { 'a': 1, 'c': 3 }
```

- `_.findKey(object, [predicate=_.identity])`

- This method is like _.find except that it returns the key of the first element predicate returns truthy for instead of the element itself.

```
var users = {
  'barney':  { 'age': 36, 'active': true },
  'fred':    { 'age': 40, 'active': false },
  'pebbles': { 'age': 1,  'active': true }
};

_.findKey(users, function(o) { return o.age < 40; });
// => 'barney' (iteration order is not guaranteed)

// The `_.matches` iteratee shorthand.
_.findKey(users, { 'age': 1, 'active': true });
// => 'pebbles'

// The `_.matchesProperty` iteratee shorthand.
_.findKey(users, ['active', false]);
// => 'fred'

// The `_.property` iteratee shorthand.
_.findKey(users, 'active');
// => 'barney'
```

- `_.omit(object, [paths])`

- This method creates an object composed of the own and inherited enumerable property paths of object that are not omitted.

```
var object = { 'a': 1, 'b': '2', 'c': 3 };

_.omit(object, ['a', 'c']);
// => { 'b': '2' }
```

- `_.unset(object, path)`

- Removes the property at path of object.

```
var object = { 'a': [{ 'b': { 'c': 7 } }] };
_.unset(object, 'a[0].b.c');
// => true

console.log(object);
// => { 'a': [{ 'b': {} }] };

_.unset(object, ['a', '0', 'b', 'c']);
// => true

console.log(object);
// => { 'a': [{ 'b': {} }] };
```

- Find more on official web page
    - https://lodash.com
- And documentation
    - https://lodash.com/docs/4.17.11

“Thank you for your time.”

–Sasa Mihaljenovic