AN INTRODUCTION TO LODASH

PROJECT #2 OF KEEPING UP WITH THE JAVASCRIPTS

WHAT IS LODASH?

Lodash is a modern JavaScript library that provides utility functions for common programming tasks, delivering modularity, performance and extras. The library uses a functional programming paradigm.

Lodash is released under the MIT license and supports modern environments.

HISTORY

Lodash was originally authored by John-David Dalton and was first released on April, 23rd 2012. It was forked from Underscore.js – another JavaScript library – and draws many of its concepts from this. Subsequently, development of Underscore.js has slowed with the core developers of Underscore devoting their energy to Lodash instead. The current stable version of Lodash is 4.17.11 and was released on September 12th 2018.

FEATURES

Lodash has many beneficial features that aid the development of JavaScript applications:

- ✓ Utilities - for simplifying common programming tasks such as determining type as well as simplifying math operations.
- ✓ Function - simplifying binding, decorating, constraining, throttling, debouncing, currying, and changing the pointer.
- ✓ String - conversion functions for performing basic string operations, such as trimming, converting to uppercase, camel case, etc.
- ✓ Array - creating, splitting, combining, modifying, and compressing
- ✓ Collection - iterating, sorting, filtering, splitting, and building
- ✓ Object - accessing, extending, merging, defaults, and transforming
- ✓ Seq - chaining, wrapping, filtering, and testing.
- ✓ Modular - so only the parts of the library that you require need to be installed or loaded into your application

Lodash has been tested in many browsers, including: Chrome 65-66, Firefox 58-59, IE 11, Edge 16, Safari 10-11, Node.js 6-10, & PhantomJS 2.1.1.

INSTALLATION

In a browser:

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

INSTALLATION

Using npm:

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

INSTALLATION

Using Node.js:

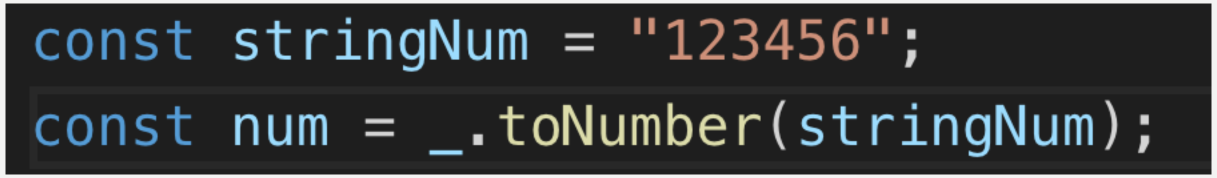
```
// Load the full build.
var _ = require('lodash');
// Load the core build.
var _ = require('lodash/core');
// Load the FP build for immutable auto-curried iteratee-first data-last methods.
var fp = require('lodash/fp');

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

// Cherry-pick methods for smaller browserify/rollup/webpack bundles.
var at = require('lodash/at');
var curryN = require('lodash/fp/curryN');
```

USAGE

After the lodash code is included in the website or installed in the application, all of its functionality – which is namespaced to the "_" character – is available for use.

Example

The "toNumber" function would be invoked as follows:



```
const stringNum = "123456";
const num = _.toNumber(stringNum);
```

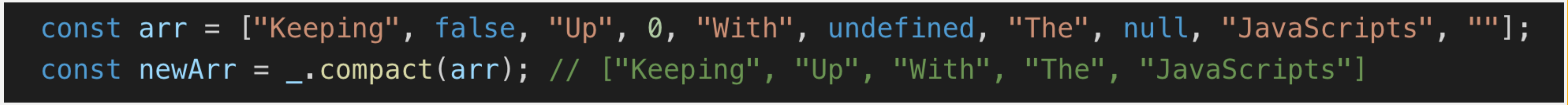
The "_" character is followed by a period, then the function that is being invoked, passing any required parameters in parentheses.

USAGE – ARRAY METHOD EXAMPLES

Lodash includes dozens of methods for working with arrays. Below are two examples of these

_.compact(array)

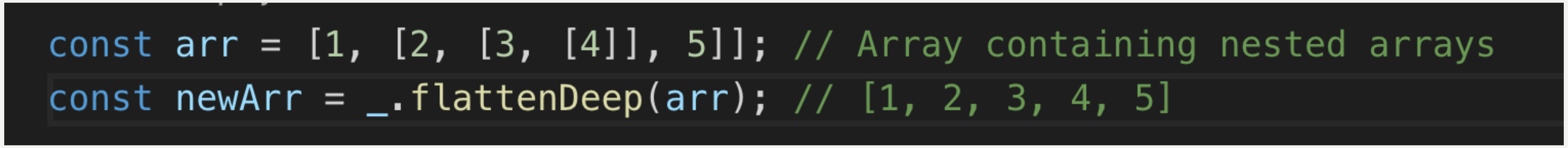
This method returns an array with all falsey values removed:



```
const arr = ["Keeping", false, "Up", 0, "With", undefined, "The", null, "JavaScripts", ""];
const newArr = _.compact(arr); // ["Keeping", "Up", "With", "The", "JavaScripts"]
```

_.flattenDeep(array)

This method recursively flattens an array, returning the values in a new array.



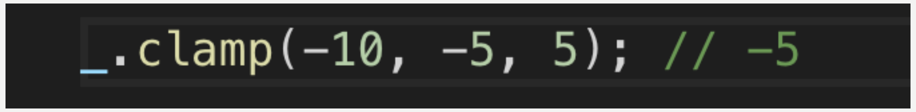
```
const arr = [1, [2, [3, [4]], 5]]; // Array containing nested arrays
const newArr = _.flattenDeep(arr); // [1, 2, 3, 4, 5]
```

USAGE – NUMBER METHOD EXAMPLES

Lodash includes three methods for working with numbers. Below are two examples of these

_.clamp(number, [lower], upper)

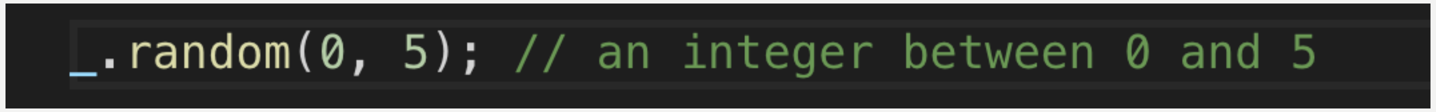
This method clamps number within the inclusive lower and upper bounds:



```
_.clamp(-10, -5, 5); // -5
```

_.random([lower=0], [upper=1], [floating])

This method returns a random number between the inclusive lower and upper bounds.



```
_.random(0, 5); // an integer between 0 and 5
```

USAGE – OBJECT METHOD EXAMPLES

Lodash includes dozens of methods for working with objects. Below are two examples of these

_.invert(object)

This method creates an object composed of the inverted keys and values of the passed object:

```javascript
const obj = { 'a': 1, 'b': 2, 'c': 3 };
const newObj = _.invert(obj); // {1: 'a', 2: 'b', 3: 'c'}
```

_.pick(object, [paths])

This method creates an object composed of the picked object properties.

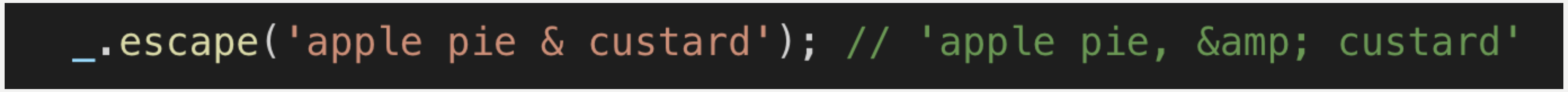
```javascript
const obj = { 'a': 1, 'b': '2', 'c': 3 };
const newObj = _.pick(obj, ['a', 'c']); // { 'a': 1, 'c': 3 }
```

USAGE – STRING METHOD EXAMPLES

Lodash includes dozens of methods for working with strings. Below are two examples of these

_.escape([string=”])

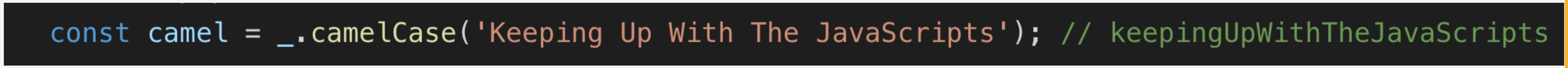
Converts the characters "&", "<", ">", """, and """ to their corresponding HTML entities.:



```
_.escape('apple pie & custard'); // 'apple pie, &amp; custard'
```

_.camelCase([string=”])

This method converts a given string to camel case.



```
const camel = _.camelCase('Keeping Up With The JavaScripts'); // keepingUpWithTheJavaScripts
```

SUMMARY

I have hardly scratched the surface of the capabilities of Lodash. However, hopefully I have demonstrated that Lodash makes coding JavaScript easier by providing functions that simplify tasks that developers are often faced with when working with arrays, numbers, objects and strings.

Even though some of the Lodash methods are now standard features within the newer specifications of JavaScript, such as ECMAScript 6 and 7, much of the functionality that Lodash supplies is simply not possible without considerable coding. Thus, Lodash provides developers with commonly needed methods and functions that give consistent results across browsers and across development teams.

FURTHER INFORMATION

Further information, code samples and documentation can be found on the Lodash website:

https://lodash.com

THANK YOU!

RICHARD CHAPPELL