transfer state forward and back

transfer state forward and back

mongo ds129053.mlab.com:29053/web322\_a71 -u <dbuser> -p <dbpassword>

mongorestore -h ds129053.mlab.com:29053 -d web322\_a71 -u jlee465 -p Vkvkdi0^

mongodb://user:password123@ds121014.mlab.com:21014/team-api-test

npm install

git init

heroku login

create heroku

git push master heroku

data-service.js --> closer

populate(join?????)

app.use(cors()); \*\*\*\*\*\*

jquery

$(function() { // this is the same as: $(document).ready(function(){

jquery-ajax

let tbody = $("#employees-table tbody");

let row = $('<tr>').append(fNameTD).append(lNameTD).append(positionTD); // create the row and append all of the TD elements.

## Introduction to Lodash

“Lodash is a toolkit of Javascript functions that provides clean, performant methods for manipulating objects and collections. It is a “fork” of the Underscore library and provides additional functionality as well as some serious performance improvements. If you aren’t using Lodash, you should be.” ~John Lindquist

As John puts it, Lodash is a toolkit or library of functions that extend some of the current working JavaScript core functionaltiy. Even with new versions of JavaScript (ie: [ES6 and Beyond)](https://en.wikipedia.org/wiki/ECMAScript#Versions) starting to catch up to all of the libraries and toolkits developed over the last decade or so, Lodash still proves itself as something many JS developers still can’t live without.

### Downloading Lodash

Like most JavaScript libraries, Lodash exists as a single, minified .js file that you can either [download](https://raw.githubusercontent.com/lodash/lodash/4.17.4/dist/lodash.core.min.js) and include in your local solution, or [reference the CDN](https://cdn.jsdelivr.net/npm/lodash@4.17.4/lodash.min.js) in your pages/views. If you’re thinking that you might want to use it in your Node.js server applications, you can do that as well [via NPM](https://www.npmjs.com/package/lodash).

### The Full Documentation

Lodash is a very large, comprehensive library and covering it in it’s entierity is well beyond the scope of this class. Instead, we will introduce some of the more interesting pieces of functionality that we’re likely to come across during this semester.

However, you should absolutely [**Bookmark the Documentation Page**](https://lodash.com/docs/) and always check it before you start working with Objects, Collections, Strings, Dates, etc. The chances are very good that there’s something in there that will help optomize your code and make your life easier.

### The Fun Stuff

As previously stated, this is only a sampling of some of the more interesting and most accessable features of the Lodash library. We will use some of these to add primitave paging and filtering functionality to last week’s **jQuery-AJAX** example.

If you’re following along with the “Code Samples” the following functionality is located in the **“week2/Lodash”**folder

### Sample Data

For the following code examples, we will assume that we have a Flintstones-inspired **users** collection that contains the following data:

**let** users **=** [

{ 'user': 'fred', 'active': **false**, 'age': 40 },

{ 'user': 'pebbles', 'active': **false**, 'age': 1 },

{ 'user': 'barney', 'active': **true**, 'age': 36 }

];

## Array Methods

### \_.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. See: <https://lodash.com/docs/4.17.4#chunk>

**let** chunk1 **=** \_.chunk(['a', 'b', 'c', 'd'], 2); *// => [['a', 'b'], ['c', 'd']]*

**let** chunk2 **=** \_.chunk(['a', 'b', 'c', 'd'], 3); *// => [['a', 'b', 'c'], ['d']]*

**let** chunk3 **=** \_.chunk(users, 2); *// objects for [['fred', 'pebbles'], 'barney']*

### \_.findIndex(array, [predicate=\_.identity], [fromIndex=0])

Returns the index of the first element **predicate** returns truthy for, instead of the element itself.

See: <https://lodash.com/docs/4.17.4#findIndex> and also <https://lodash.com/docs/4.17.4#findLastIndex> and <https://lodash.com/docs/4.17.4#find>

**let** findIndex1 **=** \_.findIndex(users, **function**(user) {

**return** user.user **==** 'fred';

}); *// => 1*

### \_.take(array, [n=1])

Creates a slice of array with n elements taken from the beginning.

See: <https://lodash.com/docs/4.17.4#take>

**let** take1 **=** \_.take(users,2) *// => objects for ['fred, pebbles']*

## Collection Methods

### \_.filter(collection, [predicate=\_.identity])

|  |  |
| --- | --- |
| Iterates over elements of **collection**, returning an array of all elements **predicate** returns truthy for. The predicate is invoked with three arguments: (value, index | key, collection). |

See: <https://lodash.com/docs/4.17.4#filter> and also: <https://lodash.com/docs/4.17.4#find>

**let** filter1 **=** \_.filter(users, **function**(user) {

**return** user.active **==** **true**;

}); *// => objects for ['barney']*

### \_.sortBy(collection, [iteratees=[\_.identity]])

Creates an array of elements, sorted in ascending order by the results of running each element in a collection thru each iteratee. This method performs a stable sort, that is, it preserves the original sort order of equal elements. The iteratees are invoked with one argument: (value).

See: <https://lodash.com/docs/4.17.4#sortBy>

**let** sortBy1 **=** \_.sortBy(users, [

**function**(user) {

**return** user.user;

}

]); *// => objects for ['barney', 'fred', 'pebbles']*

### \_.forEachRight(collection, [iteratee=\_.identity])

This method is like \_.forEach except that it iterates over elements of collection from right to left.

See: <https://lodash.com/docs/4.17.4#forEachRight> and also: <https://lodash.com/docs/4.17.4#forEach>

**let** forEachRight1 **=** [];

\_.forEachRight(users, **function**(user) {

forEachRight1.push(user);

}); *// iterates as ['barney', 'pebbles', 'fred']*

## Number Methods

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

Produces a random number between the inclusive lower and upper bounds. If only one argument is provided, a number between 0 and the given number is returned. If floating is true, or either lower or upper are floats, a floating-point number is returned instead of an integer.

See: <https://lodash.com/docs/4.17.4#random>

**let** random1 **=** \_.random(0, 5); *// => an integer between 0 and 5*

**let** random2 **=** \_.random(5); *// => also an integer between 0 and 5*

**let** random3 **=** \_.random(5, **true**); *// => a floating-point number between 0 and 5*

**let** random4 **=** \_.random(1.2, 5.2);*// => a floating-point number between 1.2 and 5.2*

## Object Methods

### \_.cloneDeep(value)

This method is like \_.clone except that it recursively clones value.

See: <https://lodash.com/docs/4.17.4#cloneDeep> and also: <https://lodash.com/docs/4.17.4#clone>

**let** cloneDeep1 **=** \_.cloneDeep(users); *// cloneDeep1[0] !== users[0]*

### \_.pick(object, [paths])

Creates an object composed of the picked object properties.

See: <https://lodash.com/docs/4.17.4#pick>

**let** pick1 **=** \_.pick(users[0], ['user', 'age']); *// => { 'user': 'fred', 'age': 40 } // note: pick !== users[0]*

## String Methods

### \_.escape([string=’’])

Converts the characters “&”, “<”, “>”, ‘”’, and “’” in string to their corresponding HTML entities.

See: <https://lodash.com/docs/4.17.4#escape> and also: <https://lodash.com/docs/4.17.4#unescape>

**let** escape1 **=** \_.escape('fred, barney, & pebbles'); *// => 'fred, barney, &amp; pebbles'*

### \_.template([string=’’], [options={}])

Creates a compiled template function that can interpolate data properties in “interpolate” delimiters, HTML-escape interpolated data properties in “escape” delimiters, and execute JavaScript in “evaluate” delimiters. Data properties may be accessed as free variables in the template. If a setting object is given, it takes precedence over [\_.templateSettings](https://lodash.com/docs/4.17.4#templateSettings) values.

See: <https://lodash.com/docs/4.17.4#template>

*// Use the "interpolate" delimiter to create a compiled template.*

**let** template1 **=** \_.template('hello <%= user %>!');

**let** template1Result **=** template1({ 'user': users[0].user }); *// => 'hello fred!'*

*// Use the HTML "escape" delimiter to escape data property values.*

**let** template2 **=** \_.template('<b><%- value %></b>');

**let** template22Result **=** template2({ 'value': '<script>' }); *// => '<b>&lt;script&gt;</b>'*

*// Use the "evaluate" delimiter to execute JavaScript and generate HTML.*

**let** template3 **=** \_.template('<ul>' **+**

'<% \_.forEach(users, function(user) { %>' **+**

'<li><%- user %></li>' **+**

'<% }); %>' **+**

'</ul>');

**let** template3Result **=** template3({ 'users': ['fred', 'barney'] }); *// => '<ul><li>fred</li><li>barney</li></ul>'*

*// Use the "evaluate" delimiter to execute JavaScript and generate HTML from our "users" collection.*

**let** template4 **=** \_.template('<ul>' **+**

'<% \_.forEach(users, function(user) { %>' **+**

'<li><%- user.user %></li>' **+**

'<% }); %>' **+**

'</ul>');

**let** template4Result **=** template4({ 'users': users }); *// => '<ul><li>fred</li><li>pebbles</li><li>barney</li></ul>'*

## Adding Sorting & Paging

Using the methods outlined above, we can very easily add “paging” and “sorting” functionality to our week1 **jquery-AJAX** example. This can be accomplished using the **.\_template()**, **.\_chunk()**, **.\_take()** and **.\_sortBy()**methods.

To see how we can use Lodash and the aforementioned properties to implement **sorting** and **paging**, open the **Lodash-AJAX** Example located in the **week2** folder. We will walk through the solution together in class.

© 2017 - Seneca School of ICT