Underscore

When migrating from Underscore to Lodash there are several differences to be aware of.
Below are some of the ones that stand out. For a more thorough guide check out lodash-codemods, eslint-plugin-lodash & Underscore to Lodash Converter.

- Underscore `_.any` is Lodash `_.some`
- Underscore `_.all` is Lodash `_.every`
- Underscore `_.compose` is Lodash `_.flowRight`
- Underscore `_.contains` is Lodash `_.includes`
- Underscore `_.each` doesn't allow exiting by returning `false` is Lodash `_.forEach`
- Underscore `_.escape` escapes backtick characters (`` ` ``), while Lodash does not
- Underscore `_.findWhere` is Lodash `_.find`
- Underscore `_.flatten` is deep by default while Lodash is shallow
- Underscore `_.groupBy` 's iteratee receives the arguments `value` , `indexNumber` , and `originalCollection` , while Lodash `_.groupBy` 's iteratee receives only the argument `value`
- Underscore `_.indexOf` with 3rd parameter `undefined` is Lodash `_.indexOf`
- Underscore `_.indexOf` with 3rd parameter `true` is Lodash `_.sortedIndexOf`
- Underscore `_.indexBy` is Lodash `_.keyBy`
- Underscore `_.invoke` is Lodash `_.invokeMap`
- Underscore `_.mapObject` is Lodash `_.mapValues`
- Underscore `_.max` combines Lodash `_.max` & `_.maxBy`
- Underscore `_.min` combines Lodash `_.min` & `_.minBy`
- Underscore `_.sample` combines Lodash `_.sample` & `_.sampleSize`
- Underscore `_.object` combines Lodash `_.fromPairs` and `_.zipObject`
- Underscore `_.omit` by a predicate is Lodash `_.omitBy`
- Underscore `_.pairs` is Lodash `_.toPairs`
- Underscore `_.pick` by a predicate is Lodash `_.pickBy`
- Underscore `_.pluck` is Lodash `_.map`
- Underscore `_.uniq` by an `iteratee` is Lodash `_.uniqBy` . `_.uniq` with the `isSorted` parameter = true is Lodash `_.sortedUniq` (or Lodash `_.sortedUniqBy` when using an iteree).
- Underscore `_.where` is Lodash `_.filter`
- Underscore `_.isFinite` doesn't align with `Number.isFinite` (e.g. `_.isFinite('1')` returns `true` in Underscore but `false` in Lodash)
- Underscore `_.matches` shorthand doesn't support deep comparisons (e.g. `_.filter(objects, { 'a': { 'b': 'c' } })`)
- Underscore ≥ 1.7 & Lodash `_.template` syntax is `_.template(string, option)(data)`
- Lodash `_.memoize` caches are `Map` like objects
- Lodash doesn't support a `context` argument for many methods in favor of `_.bind`
- Lodash supports implicit chaining, lazy chaining, & shortcut fusion
- Lodash split its overloaded `_.head` , `_.last` , `_.rest` , & `_.initial` out into `_.take` , `_.takeRight` , `_.drop` , & `_.dropRight` (i.e. `_.head(array, 2)` in Underscore is `_.take(array, 2)` in Lodash)

Backbone

Lodash works great with Backbone. It's even run against Backone's unit tests on every commit.
You can replace Underscore with Lodash in Backbone by using an AMD loader, browserify, or webpack.

Note: Lodash v4 works with Backbone ≥ v1.3.0.

- Using AMD "paths" configuration

```
require({
  'paths' {
    'backbone': 'path/to/backbone',
    'jquery': 'path/to/jquery',
    'underscore': 'path/to/lodash'
  }
}, ['backbone'], function(Backbone) {
  // use Backbone
});
```

- Using the aliasify transform for browserify by adding a section to your package.json

```
{
  "browserify": {
    "transform": ["aliasify"]
  },
  "aliasify": {
    "aliases": {
      "underscore": "lodash"
    }
  }
}
```

and executing browserify with the `aliasify` parameter

```
$ browserify entry.js --global-transform aliasify -o out.js
```

- Using the resolve.alias configuration in webpack

```
module.exports = {
  'resolve': {
    'alias': {
      'underscore': 'absolute/path/to/lodash'
    }
  }
};
```