See the v4.0.0 release notes for an overview of what's new in 4.0.0. Use lodash-migrate, lodash-codemods, & eslint-plugin-lodash to help migrate pre-4 lodash code to the latest release.

v4.17.21

***February 21, 2021* — Diff — Docs**

- Prevent command injection through `_.template`'s variable option
- Improve performance of `toNumber`, `trim` and `trimEnd` on large input strings

v4.17.20

***August 14, 2020* — Diff — Docs**

- Fixed issue with dependencies mapping
- Ensured modularized files in the lodash package are up to date

v4.17.19

***July 8, 2020* — Diff — Docs**

- Allowed `_.sortedIndexBy` methods to short-circuit for empty arrays
- Ensured `_.orderBy` accepts iteratee path arrays
- Ensured `_.isEqual` returns accurate results without depending on parameter order for circular references
- Sanitized `sourceURL` in `_.template`
- Fix prototype pollution bug in `_.zipObjectDeep`
- Note: releases of 4.17.16 to 4.17.18 of the lodash npm package were corrupted; please update to more recent versions

v4.17.15

***July 17, 2019* — Diff — Docs**

- Removed unintentionally published project helper scripts

v4.17.14

***July 10, 2019* — Diff — Docs**

- Reverted use of the `"type"` field in `package.json` of `lodash-es`

v4.17.13

***July 9, 2019* — Diff — Docs**

- Fixed missing dependency of `_createRound`

v4.17.12

***July 9, 2019* — Diff — Docs**

- Ensured `_.ceil`, `_.floor`, & `_round` handle `Infinity` consistently
- Ensured `_.debounce` clears old timers before starting new ones
- Ensured `_.mergeWith` always provides a `stack` to `customizer`

- Ensured additions to `Object.prototype` don’t break `lodash` initialization
- Ensured map and set clones contain custom properties of source values
- Fixed prototype pollution for `_.defaultsDeep`
- Fixed prototype pollution for `_.template` options

v4.17.11

***Sep. 12, 2018* — Diff — Docs**

- Ensured `_.merge` handles function properties consistently regardless of number of sources
- Ensured `Object.prototype` is not augmented by `_.merge`
- Ensured placeholder properties are set on `fp.convert()` results
- Avoided ReDoS issue in `_.words` implementation

v4.17.10

***Apr. 24, 2018* — Diff — Docs**

- Updated Lodash for better Node.js 10 support

v4.17.5

***Feb. 3, 2018* — Diff — Docs**

- Ensured `_.clone` supports subclassed arrays
- Ensured `_.cloneDeep` works with cyclical maps & sets
- Ensured `_.defaults` avoids unnecessary source property access
- Ensured `_.invert` doesn’t error on inverted values without `toString` methods
- Ensured `_.merge` & `_.mergeWith` avoid augmenting `__proto__` properties
- Ensured `_.set` supports paths with symbols
- Ensured `_.words` detects ordinals in compound words
- Ensured `fp.update` works with paths that refer to functions

v4.17.4

***Dec. 31, 2016* — Diff — Docs**

- Ensured `_.omit` with deep paths doesn’t mutate `object`

v4.17.3

***Dec. 24, 2016* — Diff — Docs**

- Added support for symbol properties to `_.isEqual`
- Ensured `getSymbols` helper only gets enumerable symbols
- Ensured `_.startsWith` avoids coercing `position` of `undefined`
- Flipped iteratee arguments for `fp.reduceRight`
- Removed the array length limit for lazy evaluation

v4.17.2

***Nov. 15, 2016* — Diff — Docs**

- Ensured `_.pick` picks keys over paths

- Ensured `_.spread` doesn’t include arguments after those spread
- Fixed `_.omit` performance regression

v4.17.1

***Nov. 14, 2016* — [Diff]() — [Docs]()**

- Ensured `_.omit` copies shallow path values by reference
- Ensured `_.pick` supports path arrays
- Ensured `_.pickBy` doesn’t treat keys with dots or brackets as deep paths

v4.17.0

***Nov. 13, 2016* — [Diff]() — [Docs]()**

- Added deep path support to `_.omit` & `_.pick`
- Ensured `fp.assignAllWith` & `fp.mergeAllWith` accept more than two sources
- Made `process.binding` detection more cautious to avoid chatty debugging

v4.16.6

***Oct. 31, 2016* — [Diff]() — [Docs]()**

- Ensured `_.xor` returns an empty array when comparing the same array

v4.16.5

***Oct. 30, 2016* — [Diff]() — [Docs]()**

- Added support for ordinal numbers to `_.words`
- Ensured `_.xor` works with more than two arrays
- Ensured `fp.convert` handles aliased & remapped methods
- Improved “fp” debugging
- Made “isType” methods resistant to `toStringTag` spoofing
- Made `_.isEmpty` exit early for nullish values
- Refined `_.isError` checks to avoid false positives for plain objects
- Simplified `_.isElement`

v4.16.4

***Oct. 6, 2016* — [Diff]() — [Docs]()**

- Added support for buffers to `_.transform`
- Added support for typed arrays to `_.isEmpty`
- Ensured `_.toString` works on an array of symbols
- Fixed `_.merge` regression with buffers
- Normalized buffers & typed arrays keys in “keys” methods
- Split `_.isArguments` out for older & newer environments

v4.16.3

***Oct. 3, 2016* — [Diff]() — [Docs]()**

- Added a `_.runInContext` check in the “fp” browser build

- Ensured `_.defaultsDeep` & `_.merge` consistently assign `undefined` values
- Fixed `_.isFunction` detection of `Proxy` in Safari 10
- Made `_.isEqual` treat buffers differently than `Uint8Array` values
- Removed es-sham requirement for older browsers

v4.16.2

***Sept. 25, 2016* — Diff — Docs**

- Fixed `_.sampleSize` performance regression
- Made "clone" methods use newer `Buffer` APIs when available

v4.16.1

***Sept. 20, 2016* — Diff — Docs**

- Fixed backtick removal of `_.escape` & `_.unescape`
- Improved parse time by 2x in V8

v4.16.0

***Sept. 19, 2016* — Diff — Docs**

- Added `fp.rangeStep` & `fp.rangeStepRight`
- Added a cache limit to internal `_.memoize` use
- Avoided V8 de-opts in `_.isElement`, `_.isObject`, & `_.isObjectLike`
- Dropped backtick encoding from `_.escape` & `_.unescape`
- Dropped testing in Node.js 0.10 & 0.12
- Ensured `__proto__` is treated as a regular key in assignments
- Fixed `_.bind` & `_.partial` performance regression
- Optimized `_.concat`

v4.15.0

***Aug. 12, 2016* — Diff — Docs**

- Added Latin Extended-A block support to `_.deburr`
- Reduced dependencies of `_.isEmpty` & `_.toNumber`

v4.14.2

***Aug. 8, 2016* — Diff — Docs**

- Ensured methods work with mocked `Date.now`, `clearTimeout`, & `setTimeout`
- Ensured paths of "set" methods overwrite primitives
- Ensured `fp.nthArg` returns a curried function
- Reduced dependencies of `_.initial`, `_.merge`, & `_.tail`
- Removed old JIT & engine bug guards

v4.14.1

***July 29, 2016* — Diff — Docs**

- Ensured paths with consecutive empty brackets or dots are parsed correctly

- Ensured `_.random` & “range” methods coerce arguments to finite numbers
- Fixed circular reference detection in `_.cloneDeep`
- Removed `global` prerequisite for `exports` detection

v4.14.0

***July 24, 2016* — [Diff]() — [Docs]()**

- Added [_.conformsTo]() & [_.defaultTo]()
- Added more “fp” aliases
- Added “All” variants of `fp/assign`, `fp/defaults`, `fp/merge`, & `fp/zip` methods
- Ensured debounced `cancel` uses `clearTimeout`
- Ensured the alias `_.first` supports shortcut fusion
- Ensured `_.assignWith` respects `customizer` results of `undefined`
- Ensured `_.divide` & `_.multiply` return `1` when no arguments are specified
- Ensured `_.isEqual` has transitive equivalence for circular references
- Fixed argument order of `fp/zipObjectDeep`
- Simplified resolving the global object
- Stopped unconditional global exports in the browser
- Made [per method packages]() zero-dependency modules
- Made wrapped function `toString` results more debuggable
- Made “flatten” methods honor `Symbol.isConcatSpreadable`
- Made `_.isEqual` treat invalid dates as equivalent
- Made `LARGE_ARRAY_SIZE` check in `stackSet` align with others
- Optimized adding values to stacks by not inspecting all key-value pairs
- Optimized “isType” methods to use faster Node.js C++ helpers when available
- Optimized `_.endsWith`, `_.negate`, & `_.startsWith`

v4.13.1

***May 23, 2016* — [Diff]() — [Docs]()**

- Ensured `_.find` & `_.findLast` provide the correct `key` param when iterating objects

v4.13.0

***May 22, 2016* — [Diff]() — [Docs]()**

- Added [_.stubArray](), [_.stubFalse](), [_.stubObject](), [_.stubString](), & [_.stubTrue]()
- Added `fromIndex` param to `_.find`, `_.findIndex`, `_.findLast`, & `_.findLastIndex`
- Ensured empty brackets & dots in paths are treated as empty property names
- Ensured `_.pullAll` works with the same value for `array` & `values`
- Ensured `_.round` works with large `precision` values
- Ensured `_.throttle` works with a system time of `0`
- Made `_.isNative` throw if `core-js` [is detected]()
- Simplified UMD exports
- Added `fp/findFrom`, `fp/findIndexFrom`, `fp/findLastFrom`, `fp/findLastIndexFrom`, `fp/indexOfFrom`, & `fp/lastIndexOfFrom`
- Fixed argument order of `fp/differenceBy`, `fp/differenceWith`, `fp/intersectionBy`, `fp/intersectionWith`, `fp/without`, `fp/unionBy`, `fp/unionWith`, `fp/xorBy`, & `fp/xorWith`

v4.12.0

***May 8, 2016* — Diff — Docs**

- Added _.toFinite
- Added iteratee arity hints to "forEach" methods
- Added support for maps & sets to `_.toPairs` & `_.toPairsIn`
- Ensured `_.merge` doesn't skip trailing function sources
- Ensured `fp/forEach`, `fp/forIn`, & `fp/forOwn` cap their iteratee arguments
- Ensured `fp/update` does not convert end of `path` to an object
- Ensured matches methods match arrays with duplicate values
- Optimized "flatten" methods
- Simplified `_.concat`
- Updated cache implementations

v4.11.2

***Apr. 21, 2016* — Diff — Docs**

- Ensured `_.pullAt` correctly sorts indexes greater than `9`
- Ensured `_.words` doesn't treat punctuation as words
- Ensured `-0` works as path arguments
- Ensured set methods like `_.uniq` & `_.xor` convert `-0` to `0`

v4.11.1

***Apr. 14, 2016* — Diff — Docs**

- Added `fp/pluck` as an alias of `fp/map`
- Ensured `_.debounce` defers invoking `func` when `leading` is `false` & `wait` is `0`

v4.11.0

***Apr. 13, 2016* — Diff — Docs**

- Added _.nth
- Added `_.matchesProperty` shorthand support to `_.sortBy`
- Added support for contractions to case methods & `_.words`
- Avoided unnecessary array cloning in `createRecurryWrapper` & `mergeData`
- Ensured `fp/over` doesn't cap its iteratee arguments
- Made `_.head` avoid accessing `array` when its length is `0`

v4.10.0

***Apr. 11, 2016* — Diff — Docs**

- Added `isIterateeCall` check to `_.split`
- Added `fp/paths` & `fp/props` as aliases of `fp/at`
- Added `fp/propEq` as an alias of `fp/matchesProperty`
- Ensured `_.debounce` queues a trailing call for subsequent debounced calls after `maxWait`
- Ensured `_.split` handles a `limit` of `0` correctly in Node.js v0.10
- Optimized handling of emojis in string methods

- Removed `fp/mapObj` & `fp/propOf` aliases

v4.9.0

***Apr. 8, 2016* — [Diff]() — [Docs]()**

- Added back `_.matchesProperty` shorthand support to "over" methods
- Ensured `_.split` works with emojis
- Fixed malformed `fp/toString` & `fp/valueOf` modules

v4.8.2

***Apr. 4, 2016* — [Diff]() — [Docs]()**

- Reverted `_.matchesProperty` shorthand support for "over" methods

v4.8.1

***Apr. 4, 2016* — [Diff]() — [Docs]()**

- Fixed typo in `fp/_falseOptions`

v4.8.0

***Apr. 4, 2016* — [Diff]() — [Docs]()**

- Added `fp/placeholder` module & `fp/__` alias
- Added `convert` to unconverted "fp" methods
- Added `_.matchesProperty` shorthand support to "over" methods
- Avoided errors in older Firefoxes when coercing `Promise` to a string
- Ensured `_.has` returns `false` for nested nullish object values
- Ensured `_.has` treats nested sparse arrays consistently
- Fixed argument order of `fp/overArgs`
- Made `_.chunk`, `_.repeat`, & `_.sampleSize` default `n` to `1` instead of `0`
- Optimized `_.matchesProperty` shorthand

v4.7.0

***Mar. 31, 2016* — [Diff]() — [Docs]()**

- Added [_.divide](), [_.flatMapDeep](), [_.flatMapDepth](), [_.meanBy](), & [_.multiply]()
- Added [fp/convert](), `fp/invokeArgs`, `fp/invokeArgsMap`, `fp/padChars`, `fp/padCharsEnd`, `fp/padCharsStart`, `fp/restFrom`, & `fp/spreadFrom`
- Added `_.entries` as an alias of `_.toPairs`
- Added `_.entriesIn` as an alias of `_.toPairsIn`
- Added several "fp" aliases
- Added support for buffers to `_.isEmpty`
- Added support for maps & sets to `_.isEmpty` & `_.size`
- Added support for deep cloning data views, maps, & sets
- Added symbol support to `_.omit`, `_.toNumber`, & `_.toPath`
- Avoided a [JIT bug]() in Safari 9 for `baseIteratee`
- Ensured array sequence methods don't error for falsey values
- Ensured `_.concat` casts non-array `array` values to arrays

- Ensured `_.has` returns `false` for nested inherited properties
- Ensured `_.isEqual` compares promises by reference
- Ensured `_.isPlainObject` returns `false` for objects with a custom `[[Prototype]]`
- Ensured `_.mergeWith` sources are cloned when `customizer` returns `undefined`
- Ensured `_.startCase` only uppercases the first character of each word
- Ensured `_.words` detects words where an all caps word is next to a capitalized word
- Fixed argument order of `fp/bind`, `fp/bindKey`, & `fp/isEqualWith`
- Fixed aliases `fp/all`, `fp/any`, & `fp/anyPass`
- Made `_.result` resolve values of functions as it deep crawls over them
- Memoized `stringToPath`
- Optimized `_.pad`, `_.padEnd`, `_.padStart`, & `_.toNumber`
- Refactored `_.debounce` to simplify, reduce timers, & fix bugs

v4.6.1

***Mar. 1, 2016* — Diff — Docs**

- Optimized `baseClone` to avoid cloning symbols for internal use

v4.6.0

***Feb. 29, 2016* — Diff — Docs**

- Added _.pullAllWith, _.update, & _.updateWith
- Added `core.min.js` & `lodash.min.js` to the “fp” module ignored list
- Ensured `_.defaultsDeep` does not overwrite regexp values
- Ensured `_.isEqual` works with maps & sets with circular references
- Ensured “fp” uses `lodash` as the default placeholder
- Ensured `stack` is popped after a recursive merge so that it doesn’t affect sibling properties
- Fixed order of arguments in “fp” docs
- Optimized `baseIsEqualDeep` by removing a typed array check & unnecessary stack creation
- Optimized `_.assign`, `_.assignIn`, & `_.intersection`
- Optimized `_.merge` to avoid deep cloning sources if a `customizer` is provided
- Removed dead code from `getNative`
- Replaced `Symbol` checks with `Symbol` methods checks

v4.5.1

***Feb. 21, 2016* — Diff — Docs**

- Added `core.min.js` & `lodash.min.js` to the npm package
- Ensured placeholders persist through more than one curried call
- Ensured `assignValue` assigns values if they aren’t the same own value
- Ensured “fp” methods avoid unnecessary cloning
- Ensured `fp/omitBy` & `fp/pickBy` provide `value` & `key` params to iteratees
- Fixed arity of `fp/orderBy`
- Used `getPrototypeOf` to set inheritance when `constructor` is a function

v4.5.0

***Feb. 17, 2016* — Diff — Docs**

- Enabled support for cloning expando properties of boolean, number, & string objects
- Ensured `fp/convert` wrappers support iteratee shorthands

v4.4.0

***Feb. 15, 2016* — [Diff]() — [Docs]()**

- Added [_.castArray]() & [_.flattenDepth]()
- Fixed argument order of `fp/inRange` & `fp/zipWith`
- Enabled `fp/convert` to auto wrap `lodash` in the browser
- Enabled `fp/convert` to work when given `lodash` with `options`
- Ensured `fp/convert` [options]() works when applied individually
- Ensured `fp/convert` works with category modules
- Ensured `_.isError` works with subclassed values
- Ensured `_.merge` deep clones array, typed-array, & plain-object sources
- Ensured a debounced `maxWait` timeout isn't processed on a leading call when `leading` is `false` & there isn't a max delay queued

v4.3.0

***Feb. 7, 2016* — [Diff]() — [Docs]()**

- Added [_.isArrayBuffer](), [_.isBuffer](), [_.isMap](), [_.isSet](), [_.isWeakMap](), & [_.isWeakSet]()
- Added [options]() param to `fp/convert`
- Ensured default placeholder values are set for relevant method modules
- Ensured `_.add` & `_.subtract` return `0` when no arguments are given
- Ensured `fp/gt`, `fp/gte`, `fp/lt`, & `fp/lte` don't have `rearg` applied
- Improved accuracy of `_.debounce` & `_.throttle`

v4.2.1

***Feb. 3, 2016* — [Diff]() — [Docs]()**

- Added remap of `fp/trim` as `fp/trimChars` for `chars` param support
- Ensured wrapped `_.bind`, `_.curry`, & `_.partial` support placeholders
- Ensured `fp/_baseConvert` uses `_.spread` for `_.partial` wrapper
- Ensured `fp/add` & `fp/subtract` don't have `rearg` applied

v4.2.0

***Feb. 2, 2016* — [Diff]() — [Docs]()**

- Added `fp/assoc` & `fp/assocPath` as aliases of `fp/set`
- Added `fp/dissoc` & `fp/dissocPath` as aliases of `fp/unset`
- Added `start` param to `_.spread`
- Ensured `_.attempt` preserves custom errors
- Ensured `fp/partial` & `fp/partialRight` accept an `args` param
- Ensured `fp/unset` is immutable
- Ensured `_.iteratee` clones sources for `_.matchesProperty` shorthand
- Made `_.flatMap` a "Collection" method
- Removed `global` references from the npm package

v4.1.0

***Jan. 29, 2016* — [Diff] — [Docs]**

- Added [_.invertBy] & [_.zipObjectDeep]
- Added cherry pickable “fp” method modules to the npm package
- Ensured `fp/convert` works with aliases
- Ensured `_.clone` & `_.cloneDeep` work on prototype objects
- Ensured `_.mergeWith` overwrites primitives with source object clones
- Ensured `_.sum` & `_.sumBy` return `0` for empty arrays
- Optimized `_.isEmpty` for objects
- Fixed argument order of `fp/assign`, `fp/defaults`, & `fp/merge`

v4.0.1

***Jan. 25, 2016* — [Diff] — [Docs]**

- Added `_.matches` to the core build
- Added placeholder support to “fp” methods
- Added support for keycap emojis
- Ensured `_.concat` returns an empty array for nullish values
- Ensured `_.each` & `_.eachRight` aliases have the correct chain behavior
- Ensured `_.defaultsDeep` doesn’t convert function properties to objects
- Ensured `_.fromPairs` can consume results of `_.toPairs`
- Ensured `_.isEqual` compares objects unordered
- Ensured `_.noConflict` restores `_` only if `lodash` is the current value
- Ensured `_.words` captures all-caps words
- Ensured `_.words` treats all-lower & all-upper postfixes as separate words
- Fixed “fp” mapping of several methods
- Made `_.omitBy` & `_.pickBy` provide a `key` param to iteratees

v4.0.0

***Jan. 12, 2016* — [Diff] — [Docs]**

Compatibility Warnings

- Removed Bower & Component package support in favor of npm
- Dropped [IE 6-8 support]
- Use [es5-shim], & optionally [es6-shim], to enable support
- Made `_#times`, `_#forEach`, `_#forIn`, `_#forOwn`, & their right-forms implicitly end chain sequences

```
var wrapped = _([1, 2, 3]);

// in 3.10.1
wrapped.forEach(function(n) { console.log(n); });
// ➜ returns the lodash wrapper without logging until `value` is called
wrapped.forEach(function(n) { console.log(n); }).value();
// ➜ logs each value from left to right and returns the array
// in 4.0.0
wrapped.forEach(function(n) { console.log(n); });
// ➜ logs each value from left to right and returns the array
```

- Removed category names from module paths

```
// in 3.10.1
var chunk = require('lodash/array/chunk');

// in 4.0.0
var chunk = require('lodash/chunk');
```

- Removed `_.pluck` in favor of `_.map` with iteratee shorthand

```
var objects = [{ 'a': 1 }, { 'a': 2 }];

// in 3.10.1
_.pluck(objects, 'a'); // ➜ [1, 2]
_.map(objects, 'a'); // ➜ [1, 2]

// in 4.0.0
_.map(objects, 'a'); // ➜ [1, 2]
```

- Removed `thisArg` params from most methods because they were largely unused, complicated implementations, & can be tackled with `_.bind`, `Function#bind`, or arrow functions

```
var objects = [{ 'a': 1 }, { 'a': 2 }];
var context = { 'b': 5 };

function callback(item) {
  return item.a + this.b;
}

// in 3.10.1
_.map(objects, callback, context);

// in 4.0.0
_.map(objects, _.bind(callback, context));
```

- Split `_.max` & `_.min` into [_.maxBy](#) & [_.minBy](#)

```
var array = [1, 2, 3],
    objects = [{ 'a': 1 }, { 'a': 2 }];

// in 3.10.1
_.max(array); // ➜ 3
_.max(objects, 'a'); // ➜ { 'a': 2 }
```

```
_.min(array); // ➜ 1
_.min(objects, 'a'); // ➜ { 'a': 1 }

// in 4.0.0
_.max(array); // ➜ 3
_.maxBy(objects, 'a'); // ➜ { 'a': 2 }

_.min(array); // ➜ 1
_.minBy(objects, 'a'); // ➜ { 'a': 1 }
```

- Method removals
  - Removed `_.support`
  - Removed `_.findWhere` in favor of `_.find` with iteratee shorthand
  - Removed `_.where` in favor of `_.filter` with iteratee shorthand
  - Removed `_.pluck` in favor of `_.map` with iteratee shorthand
- Method renames
  - Renamed `_.first` to `_.head`
  - Renamed `_.indexBy` to `_.keyBy`
  - Renamed `_.invoke` to `_.invokeMap`
  - Renamed `_.modArgs` to `_.overArgs`
  - Renamed `_.padLeft` & `_.padRight` to `_.padStart` & `_.padEnd`
  - Renamed `_.pairs` to `_.toPairs`
  - Renamed `_.rest` to `_.tail`
  - Renamed `_.restParam` to `_.rest`
  - Renamed `_.sortByOrder` to `_.orderBy`
  - Renamed `_.trimLeft` & `_.trimRight` to `_.trimStart` & `_.trimEnd`
  - Renamed `_.trunc` to `_.truncate`
- Split out methods
  - Split `_.assign` & `_.assignIn` into `_.assignWith` & `_.assignInWith`
  - Split `_.clone` & `_.cloneDeep` into `_.cloneWith` & `_.cloneDeepWith`
  - Split `_.indexOf` & `_.lastIndexOf` into `_.sortedIndexOf` & `_.sortedLastIndexOf`
  - Split `_.invert` into `_.invertBy` (see v4.1.0)
  - Split `_.isEqual` into `_.isEqualWith`
  - Split `_.isMatch` into `_.isMatchWith`
  - Split `_.max` & `_.min` into `_.maxBy` & `_.minBy`
  - Split `_.merge` into `_.mergeWith`
  - Split `_.omit` & `_.pick` into `_.omitBy` & `_.pickBy`
  - Split `_.sample` into `_.sampleSize`
  - Split `_.sortedIndex` into `_.sortedIndexBy`
  - Split `_.sortedLastIndex` into `_.sortedLastIndexBy`
  - Split `_.sum` into `_.sumBy`
  - Split `_.uniq` into `_.sortedUniq`, `_.sortedUniqBy`, & `_.uniqBy`
  - Split `_.zipObject` into `_.fromPairs`
- Absorbed `_.sortByAll` into `_.sortBy`
- Changed the category of `_.at` to “Object”
- Changed the category of `_.bindAll` to “Util”

- Changed `_.matchesProperty` shorthand to an array of `[path, srcValue]`
- Enabled `_.merge` to assign `undefined` if the destination property doesn't exist
- Made "By" methods like `_.groupBy` & `_.sortBy` provide a single param to iteratees
- Made `_.add`, `_.max`, `_.min`, & `_.sum` no longer coerce values to numbers
- Made `_.capitalize` uppercase the first character & lowercase the rest (see _.upperFirst)
- Made `_.eq` its own method instead of an alias for `_.isEqual`
- Made `_.functions` return only own method names
- Made `_.max` & `_.min` return `undefined` when passed an empty array
- Made `_.words` chainable by default
- Moved `./lodash.js` to `./dist/lodash.js` in the `master` branch
- Moved `./index.js` to `./lodash.js` in the `npm` branch
- Removed `isDeep` params from `_.clone` & `_.flatten`
- Removed `multiValue` param from `_.invert`
- Removed support for binding all methods by default from `_.bindAll`
- Removed `func`-first param signature from `_.before` & `_.after`

Low Risk Compatibility Warnings

- Dropped boolean `options` param support in `_.debounce`, `_.mixin`, & `_.throttle`
- Dropped support for boolean `orders` param in `_.orderBy`
- Made `_.escapeRegExp` align to the defunct ES7 proposal
- Made `_.max`, `_.min`, & `_.sum` support arrays only
- Removed legacy `options` param signature from `_.template`

Notable Changes

- Core build
- 4 kB (gzipped) core build (64 methods; Backbone ≥ v1.3.0 compatible)
  `_.assignIn`, `_.before`, `_.bind`, `_.chain`, `_.clone`, `_.compact`, `_.concat`, `_.create`, `_.defaults`, `_.defer`, `_.delay`, `_.each`, `_.escape`, `_.every`, `_.filter`, `_.find`, `_.first`, `_.flatten`, `_.flattenDeep`, `_.forEach`, `_.has`, `_.head`, `_.identity`, `_.indexOf`, `_.isArguments`, `_.isArray`, `_.isBoolean`, `_.isDate`, `_.isEmpty`, `_.isEqual`, `_.isFinite`, `_.isFunction`, `_.isNaN`, `_.isNull`, `_.isNumber`, `_.isObject`, `_.isRegExp`, `_.isString`, `_.isUndefined`, `_.iteratee`, `_.keys`, `_.last`, `_.map`, `_.matches`, `_max`, `_.min`, `_.mixin`, `_.negate`, `_.noConflict`, `_.noop`, `_.once`, `_.pick`, `_.reduce`, `_.result`, `_.size`, `_.slice`, `_.some`, `_.sortBy`, `_.tap`, `_.thru`, `_.toArray`, `_.uniqueId`, `_#value`, & `_.values`
- Reduced functionality:
  - Cloning only supports arrays & `Object` objects
  - Iteratee shorthand for `_.matchesProperty` is removed
  - Lazy evaluation is removed
  - Placeholder support is removed
  - Support for deep property paths is removed
  - Support for maps, sets, & typed arrays is removed
- Added 80 methods
- 22 array methods:
  _.concat, _.differenceBy, _.differenceWith, _.fromPairs, _.intersectionBy, _.intersectionWith, _.join, _.pullAll, _.pullAllBy, _.reverse, _.sortedIndexBy, _.sortedIndexOf, _.sortedLastIndexBy, _.sortedLastIndexOf, _.sortedUniq, _.sortedUniqBy, _.unionBy, _.unionWith, _.uniqBy, _.uniqWith, _.xorBy, & _.xorWith

- 18 lang methods:
  _.cloneDeepWith, _.cloneWith, _.eq, _.isArrayLike, _.isArrayLikeObject, _.isEqualWith, _.isInteger, _.isLength, _.isMatchWith, _.isNil, _.isObjectLike, _.isSafeInteger, _.isSymbol, _.toInteger, _.toLength, _.toNumber, _.toSafeInteger, & _.toString

- 13 object methods:
  _.assignIn, _.assignInWith, _.assignWith, _.functionsIn, _.hasIn, _.invoke, _.mergeWith, _.omitBy, _.pickBy, _.setWith, _.toPairs, _.toPairsIn, & _.unset

- 8 string methods:
  _.lowerCase, _.lowerFirst, _.replace, _.split, _.upperCase, _.upperFirst, _.toLower, & _.toUpper

- 8 utility methods:
  _.cond, _.conforms, _.nthArg, _.over, _.overEvery, _.overSome, _.rangeRight, & _.toPath

- 4 math methods:
  _.maxBy, _.mean, _.minBy, & _.sumBy

- 2 collection methods:
  _.flatMap (see v4.2.0) & _.sampleSize

- 2 function methods:
  _.flip & _.unary

- 2 number methods:
  _.clamp & _.subtract

- 1 chain method:
  _#next

- Added 3 aliases
- Added `_.extend` as an alias of `_.assignIn`
- Added `_.extendWith` as an alias of `_.assignInWith`
- Added `_.first` as an alias of `_.head`
- Removed 17 aliases
- Removed `_.all` in favor of `_.every`
- Removed `_.any` in favor of `_.some`
- Removed `_.backflow` in favor of `_.flowRight`
- Removed `_.callback` in favor of `_.iteratee`
- Removed `_.collect` in favor of `_.map`
- Removed `_.compose` in favor of `_.flowRight`
- Removed `_.contains` in favor of `_.includes`
- Removed `_.detect` in favor of `_.find`
- Removed `_.foldl` in favor of `_.reduce`
- Removed `_.foldr` in favor of `_.reduceRight`
- Removed `_.include` in favor of `_.includes`
- Removed `_.inject` in favor of `_.reduce`
- Removed `_.methods` in favor of `_.functions`
- Removed `_.object` in favor of `_.fromPairs` or `_.zipObject`
- Removed `_#run` in favor of `_#value`
- Removed `_.select` in favor of `_.filter`

- Removed `_.unique` in favor of `_.uniq` instead

- Performance improvements

- Enabled shortcut fusion for `_.at`, `_.find` & `_.findLast`
- Optimized match methods to avoid deep crawling if `object` & `source` identical
- Optimized circular reference searches
- Optimized `_.isEqual` to avoid stack crawls when arrays or objects are different sizes

- Emoji support

- Added support for astral symbols, combining diacritical marks, dingbats, regional indicator symbols, unicode modifiers, variation selectors, & zero-width-joiners to string methods

- Functional goodies

- Added _.cond, _.conforms, _.flip, _.nthArg, _.over, _.overEvery, _.overSome, & _.unary
- Moved `lodash-fp` into `lodash` as `require('lodash/fp')` for immutable auto-curried iteratee-first data-last methods

Other Changes

- Added `clear` method to `_.memoize.Cache`
- Added `flush` method to debounced & throttled functions
- Added support for maps, sets, & symbols to `_.clone`, `_.isEqual`, & `_.toArray`
- Added support for array buffers to `_.isEqual`
- Added support for converting iterators to `_.toArray`
- Added support for deep paths to `_.zipObject`
- Changed UMD to export to `window` or `self` when available regardless of other exports
- Enabled `_.flow` & `_.flowRight` to accept an array of functions
- Ensured "Collection" methods treat functions as objects
- Ensured debounce `cancel` clears `args` & `thisArg` references
- Ensured `_.add` & `_.sum` don't skip `NaN` values
- Ensured `_.assign`, `_.defaults`, & `_.merge` coerce `object` values to objects
- Ensured `_.bindKey` bound functions call `object[key]` when called with the `new` operator
- Ensured `_.clone` treats generators like functions
- Ensured `_.clone` produces clones with the source's `[[Prototype]]`
- Ensured `_.defaults` assigns properties that shadow `Object.prototype`
- Ensured `_.defaultsDeep` doesn't merge a string into an array
- Ensured `_.defaultsDeep` & `_.merge` don't modify sources
- Ensured `_.defaultsDeep` works with circular references
- Ensured `_.isFunction` returns `true` for generator functions
- Ensured `_.keys` skips "length" on strict mode `arguments` objects in Safari 9
- Ensured `_.merge` assigns typed arrays directly
- Ensured `_.merge` doesn't convert strings to arrays
- Ensured `_.merge` merges plain-objects onto non plain-objects
- Ensured `_#plant` resets iterator data of cloned sequences
- Ensured `_.random` swaps `min` & `max` if `min` is greater than `max`
- Ensured `_.range` preserves the sign of `start` of `-0`
- Ensured `_.reduce` & `_.reduceRight` use `getIteratee` in their array branch
- Fixed rounding issue with the `precision` param of `_.floor`
- Made `_(...)` an iterator & iterable
- Made `_.drop`, `_.take`, & right forms coerce `n` of `undefined` to `0`

v3.10.1

***Aug. 4, 2015* — Diff — Docs**

- Ensured `func` is a function before calling `getData`
- Ensured `_.clone` provides the correct number of arguments to `customizer`
- Ensured `_#reverse` doesn’t modify the original array when called after `_#slice`

v3.10.0

***June 30, 2015* — Diff — Docs**

- Added `_.ceil`, `_.defaultsDeep`, `_.floor`, `_.modArgs`, & `_.round`
- Ensured `_.bind` works with all built-in constructors
- Ensured `_#concat` treats sparse arrays as dense
- Ensured `_.chunk` floors `size` values
- Ensured `_.debounce` & `_.throttle` reset `lastCall` after cancelling
- Ensure `_.flow` & `_.flowRight` work with functions combined with `_.first`
- Ensured `_.indexOf` returns `-1` for unmatched binary searches
- Ensured `_.noConflict` operates on `root` & not `context`
- Made `_.escapeRegExp` more robust
- Made `_.sortByOrder` support orders of “asc” & “desc”
- Optimized `_.flatten`, `_.max`, & `_.min`
- Removed fallbacks for `_.isArguments`, `_.isFunction`, & `_.isPlainObject`
- Simplified lazy evaluation support
- Simplified `_.isElement`, `_.isFinite`, `_.isNative`, `_.now`, `_.parseInt`, & `_.sum`

v3.9.3

***May 26, 2015* — Diff — Docs**

- Made `parseFloat` assigned from the `context` param of `_.runInContext`
- Ensured `_.set` handles non-index property names that start with numbers correctly

v3.9.2

***May 24, 2015* — Diff — Docs**

- Made `isLaziable` work with wrapped lodash methods
- Optimized an early exit case in `_.isEqual`
- Optimized `_.sample`

v3.9.0

***May 19, 2015* — Diff — Docs**

- Added `_.gt`, `_.gte`, `_.lt`, & `_.lte`
- Added support for an `ImmutableMap` to `_.memoize.Cache`
- Avoided using `require` in source comments to improve browserify build times
- Ensured `baseCreate` works in ExtendScript
- Ensured `customizer` results are respected by `_.isEqual`
- Ensured `LodashWrapper.prototype.thru` exists before creating a wrapper in `flow`

- Ensured `_.bind` works with ES2015 class constructors
- Ensured `_.get` can return `null` values
- Ensured `_.intersection` works with a single array
- Ensured `_.has` treats sparse arrays as dense
- Ensured `_.merge` skips `undefined` array values if a destination value exists
- Made `null` sort behind `undefined` & `NaN`
- Made `_.eq` an alias of `_.isEqual`
- Optimized object comparisons in `_.isEqual`
- Optimized `_.max` & `_.min` when invoked with iteratees
- Optimized `_.pullAt` & `_.remove`
- Used `hasOwnProperty` for the creation of `reIsNative` to avoid issues with core-js

v3.8.0

***May 1, 2015* — Diff — Docs**

- Added `_.mapKeys`, `_.unzipWith`, & `_.zipWith`
- Made `_.difference`, `_.intersection`, `_.uniq`, & `_.xor` accept array-like values
- Ensured empty brackets are ignored by `_.get` & `_.set`
- Ensured `baseAt`, `basePullAt`, & `pullAt` handle nullish values
- Ensured `baseGet` only returns `undefined` for incomplete paths
- Ensured `_.padLeft` & `_.padRight` handle empty strings correctly
- Made `_.isEqual` treat `-0` & `0` as equivalent
- Narrowed bitmask checks in `_.flow` & `_.flowRight`
- Optimized lazy `slice` for `start` of `0`
- Optimized “flatten” methods
- Restricted `Object.assign` use to strict mode only

v3.7.0

***Apr. 15, 2015* — Diff — Docs**

- Added `_.get`, `_.method`, `_.methodOf`, & `_.set`
- Avoided a JIT bug in Safari on at least iOS 8.1-8.3 ARM64
- Ensured `_.intersection` of a single array returns an empty array
- Ensured `_.remove` mutates `array` after the `predicate` pass
- Ensured methods like `_.has` & `_.get` can access index values of strings
- Made `createAssigner` & `_.sortByAll` use `_.restParam`
- Made `_.add` coerce `augend` & `addend` params to numbers
- Made `_.assign` use built-in `Object.assign` when available
- Made `_.inRange` swap `start` & `end` params when `start` is greater than `end`
- Added deep path support to methods like `_.has`, `_.get`, `_.callback`, `_.invoke`, `_.matchesProperty`, `_.method`, `_.methodOf`, `_.property`, `_.propertyOf`, `_.result`, & `_.set`

v3.6.0

***Mar. 25, 2015* — Diff — Docs**

- Added `_.restParam`

- Added `isIterateeCall` guards to `_.every`, `_.includes`, & `_.some`
- Added support for `iteratee` & `thisArg` params to `_.sum`
- Added support for shortcut fusion optimizations to `_.flow` & `_.flowRight`
- Ensured lodash bundled by r.js runs in a web worker
- Ensured `_.deburr` removes combining diacritical marks
- Ensured `_.difference` is based on the values of the first param only
- Expanded metadata optimizations to more combinations of curried functions
- Made `_.matches` work with non-plain objects & match inherited properties
- Optimized `_.findLast`, `_.reduce`, & `_.reduceRight`
- Reduced code duplication with internal method creator functions

v3.5.0

***Mar. 8, 2015* — [Diff]() — [Docs]()**

- Added string `replace` & `split` chaining methods
- Ensured lazy `drop` works when applied after `filter`
- Optimized curried method performance

v3.4.0

***Mar. 6, 2015* — [Diff]() — [Docs]()**

- Added `_.add`, `_.sortByOrder`, & `_.sum`
- Adjusted `root` to work when bundled by webpack & running in a web worker
- Ensured lazy chaining works with combinations of `drop` & `dropWhile`
- Ensured `_.defaults` works as an iteratee for `_.reduce`
- Reduced side effects of shortcut fusion
- Optimized `baseFlatten`, `baseIndexOf`, `indexOfNaN`, `_.flow`, & `_.flowRight`
- Optimized lazy evaluation for iteratees with one param

v3.3.1

***Feb. 24, 2015* — [Diff]() — [Docs]()**

- Ensured lazy `takeWhile` works with lazy `reverse` & `last`
- Ensured `wait` defaults to `0` for `_.debounce` & `_.throttle`
- Ensured `isIterateeCall` doesn’t error if `index` is missing a `toString` method
- Optimized `_.difference`, `_.intersection`, & `_.uniq` in Firefox

v3.3.0

***Feb. 20, 2015* — [Diff]() — [Docs]()**

- Added `_.inRange`
- Added links to each method’s [npm package]() to the [documentation]()
- Enabled `_.clone` to work with more truthy `isDeep` values
- Ensured each lodash wrapper module sets up their inheritance
- Ensured `isIterateeCall` works with `NaN` values
- Ensured `_.merge` avoids iterating string `object` or `source` params
- Optimized `_.attempt`
- Reduced nested dependencies of `_.template`

v3.2.0

***Feb. 12, 2015* — [Diff] — [Docs]**

- Added `_.fill`, `_.matchesProperty`, & `_.spread`
- Added `_#commit` & `_#plant`
- Added support for lazy `_.compact` & `_.toArray`
- Enabled `_.attempt` to provide additional arguments to `func`
- Ensured `_.flow` returns an identity function when called without arguments
- Ensured `_#reverse` tracks `__chain__` values
- Ensured lazy `_.slice` handles floating-point `start` & `end` params
- Fixed lazy `_.slice` when used after `_.filter`
- Made `_#run` an alias of `_#value`

v3.1.0

***Feb. 3, 2015* — [Diff] — [Docs]**

- Added `_.startCase`
- Ensured `isIterateeCall` works correctly with objects

v3.0.1

***Jan. 30, 2015* — [Diff] — [Docs]**

- Ensured `_.slice` coerces floating-point `start` & `end` params to integers
- Fixed lazy `initial`, `rest`, & `reverse`
- Fixed `_.merge` regression with DOM elements

v3.0.0

***Jan. 26, 2015* — [Diff] — [Docs]**

Compatibility Warnings

- Made chaining lazy, that is, execution is deferred until `_#value` is implicitly or explicitly called

```
var wrapped = _([1, 2, 3]);

// in 2.4.1
wrapped.forEach(function(n) { console.log(n); });
// ➜ logs each value from left to right and returns the lodash wrapper

// in 3.0.0
wrapped.forEach(function(n) { console.log(n); });
// ➜ returns the lodash wrapper without logging until `value` is called
wrapped.forEach(function(n) { console.log(n); }).value();
// ➜ logs each value from left to right and returns the array
```

- Ensured each segment of a chain sequence may be repeatly applied

```
var array = [1],
    wrapped = _(array);

// in 2.4.1
var a = wrapped.push(2), // pushes `2` to `array`
    b = wrapped.push(3); // pushes `3` to `array`

a.value(); // → returns `array`; [1, 2, 3]
b.value(); // → returns `array`; [1, 2, 3]

// in 3.0.0
var a = wrapped.push(2), // creates a lazy sequence to push `2` to `array`
    b = wrapped.push(3); // creates a lazy sequence to push `3` to `array`

a.value(); // → pushes `2` to `array` and returns `array`; [1, 2]
b.value(); // → pushes `3` to `array` and returns `array`; [1, 2, 3]
a.value(); // → pushes `2` to `array` and returns `array`; [1, 2, 3, 2]
b.value(); // → pushes `3` to `array` and returns `array`; [1, 2, 3, 2, 3]

// use `_#commit` to commit a sequence and continue chaining
var a = wrapped.push(2).commit(), // pushes `2` to `array`
    b = wrapped.push(3).commit(); // pushes `3` to `array`

a.value(); // → returns `array`; [1, 2, 3]
b.value(); // → returns `array`; [1, 2, 3]
```

- Made `_.flatten` shallow by default & remove `callback` support

```
var array = [1, [[2], 3]],
    objects = [{ 'a': [1] }, { 'a': [2, 3] }];

// in 2.4.1
_.flatten(array); // → [1, 2, 3]
_.flatten(objects, 'a'); // [1, 2, 3]

// in 3.0.0
_.flatten(array); // → [1, [2], 3]
_.flattenDeep(array); // → [1, 2, 3]
_(objects).pluck('a').flatten().value(); // [1, 2, 3]
```

- Removed the `data` parameter from `_.template`

```
var string = '<%= o.a %>',
    options = { 'variable': 'o' },
    data = { 'a': 'b' };

// in 2.4.1
_.template(string, data, options); // → 'b'
```

```
// in 3.0.0
_.template(string, options)(data); // ➜ 'b'
```

- Split `_.first` & `_.last` into `_.take`, `_.takeWhile`, `_.takeRight`, & `_.takeRightWhile`

```
var array = [1, 2, 3],
    lessThanTwo = function(value) { return value < 2; },
    greaterThanTwo = function(value) { return value > 2; };

// in 2.4.1
_.first(array); // ➜ 1
_.first(array, 2); // ➜ [1, 2]
_.first(array, lessThanTwo);   // ➜ [1]

_.last(array); // ➜ 3
_.last(array, 2); // ➜ [2, 3]
_.last(array, greaterThanTwo); // ➜ [3]

// in 3.0.0
_.first(array); // ➜ 1
_.take(array, 2); // ➜ [1, 2]
_.takeWhile(array, lessThanTwo); // ➜ [1]

_.last(array); // ➜ 3
_.takeRight(array, 2); // ➜ [2, 3]
_.takeRightWhile(array, greaterThanTwo); // ➜ [3]
```

- Split `_.initial` & `_.rest` into `_.dropRight`, `_.dropRightWhile`, `_.drop`, & `_.dropWhile`

```
var array = [1, 2, 3],
    lessThanTwo = function(value) { return value < 2; },
    greaterThanTwo = function(value) { return value > 2; };

// in 2.4.1
_.initial(array); // ➜ [1, 2]
_.initial(array, 2); // ➜ [1]
_.initial(array, greaterThanTwo); // ➜ [1, 2]

_.rest(array); // ➜ [2, 3]
_.rest(array, 2); // ➜ [3]
_.rest(array, lessThanTwo); // ➜ [2, 3]

// in 3.0.0
_.initial(array); // ➜ [1, 2]
_.dropRight(array, 2); // ➜ [1]
_.dropRightWhile(array, greaterThanTwo); // ➜ [1, 2]

_.rest(array); // ➜ [2, 3]
```

```
_.drop(array, 2); // ➜ [3]
_.dropWhile(array, lessThanTwo); // ➜ [2, 3]
```

- Aligned `_.isFinite` & `_.keys` with ES2015
- Changed the category of `_.clone` & `_.cloneDeep`, & "isType" methods from "Object" to "Lang"
- Changed the category of `_.escape`, `_.template`, & `_.unescape` from "Util" to "String"
- Changed the category of `_.range` from "Array" to "Util"
- Changed the category of `_.toArray` from "Collection" to "Lang"
- Made method categories singular
- Made "Object" methods coerce primitives to objects
- Made `_.clone` & `_.cloneDeep` return a new object for unsupported types
- Made `_.findWhere` its own method instead of an alias for `_.find`
- Made `_.max` & `_.min` non-chainable by default
- Made `_.memoize` caches have the `Map` interface of `delete`, `get`, `has`, & `set`
- Made `_.unzip` its own method instead of an alias for `_.zip`
- Moved `./dist/lodash.js` to `./lodash.js` in the `master` branch
- Moved `./dist/lodash.js` to `./lodash.js` in the `npm` branch
- Moved `./dist/lodash.compat.js` to the `lodash/lodash-compat` repository
- Moved support for sorting by multiple properties from `_.sortBy` to `_.sortByAll`
- Removed result sorting from `_.functions`
- Removed the `underscore` build
- Renamed `_.createCallback` to `_.callback`
- Renamed `_.support.argsClass` to `_.support.argsTag`
- Renamed `_.support.nodeClass` to `_.support.nodeTag`
- Restricted `_.mixin` to iterating only own properties of `source` objects
- Expanded the number of arguments provided to `customizer` callbacks of `_.assign`, `_.clone`, `_.cloneDeep`, `_.isEqual`, & `_.merge`

Low Risk Compatibility Warnings

- Added customizable argument placeholder support to `_.bind`, `_.bindKey`, `_.curry`, `_.curryRight`, `_.partial`, & `_.partialRight`
- Added support for matching `NaN` to `_.difference`, `_.includes`, `_.indexOf`, `_.intersection`, `_.lastIndexOf`, `_.union`, `_.uniq`, & `_.xor`
- Ensured `_.assign` & `_.merge` don't assign `customizer` results if it's unchanged
- Ensured `_.mixin` doesn't extend lodash when given an empty `options` object
- Ensured `_.sortedIndex` returns values that align with the sort order of `_.sortBy`
- Ensured functions of `_.matches` return `true` when comparing empty sources
- Ensured functions of `_.matches` perform own property checks on objects
- Made `_.chain` use an existing wrapper if available instead of creating a new wrapper
- Removed the `argCount` parameter from `_.callback`

Notable Changes

- Added 47 methods

- 17 string methods:
  `_.camelCase`, `_.capitalize`, `_.deburr`, `_.endsWith`, `_.escapeRegExp`, `_.kebabCase`, `_.pad`, `_.padLeft`, `_.padRight`, `_.repeat`, `_.snakeCase`, `_.startsWith`, `_.trim`, `_.trimLeft`, `_.trimRight`, `_.trunc`, & `_.words`

- 11 array methods:
  `_.chunk`, `_.dropRight`, `_.dropRightWhile`, `_.dropWhile`, `_.flattenDeep`, `_.pullAt`, `_.slice`, `_.sortedLastIndex`, `_.takeRight`, `_.takeRightWhile`, & `_.takeWhile`

- 6 function methods:
  `_.ary`, `_.before`, `_.curryRight`, `_.flow`, `_.negate`, & `_.rearg`

- 5 lang methods:
  `_.isError`, `_.isMatch`, `_.isNative`, `_.isTypedArray`, & `_.toPlainObject`

- 3 utility methods:
  `_.attempt`, `_.matches`, & `_.propertyOf`

- 2 collection methods:
  `_.partition` & `_.sortByAll`

- 2 object methods:
  `_.keysIn` & `_.valuesIn`

- 1 chain method:
  `_.thru`

- Added 5 aliases

- Added `_.backflow` & `_.compose` as aliases of `_.flowRight`
- Added `_.contains` as an alias of `_.includes`
- Added `_.iteratee` as an alias of `_.callback`
- Added `_#toJSON` as an alias of `_#valueOf`

- Performance improvements

- Improved overall performance 20-40%
- Method chaining supports lazy evaluation
- Methods with support for shortcut fusion:
  `_.drop`, `_.dropRight`, `_.dropRightWhile`, `_.dropWhile`, `_.filter`, `_.first`, `_.initial`, `_.last`, `_.map`, `_.pluck`, `_.reject`, `_.rest`, `_.reverse`, `_.slice`, `_.take`, `_.takeRight`, `_.takeRightWhile`, `_.takeWhile`, & `_.where`
- Other optimized methods:
  `_.bind`, `_.clone`, `_.cloneDeep`, `_.compact`, `_.compose`, `_.contains`, `_.difference`, `_.escape`, `_.flatten`, `_.invoke`, `_.isEqual`, `_.isObject`, `_.matches`, `_.max`, `_.min`, `_.partial`, `_.shuffle`, `_.unescape`, `_.uniq`, `_.without`, & `_.zip`

- Functional goodies

- Added `_.ary`, `_.curryRight`, `_.flow`, & `_.rearg`
- Added placeholder support to `_.bind`, `_.bindKey`, `_.curry`, `_.curryRight`, `_.partial`, & `_.partialRight`
- Methods that work as an iteratee for `_.map`, & the like, out of the box:
  `_.ary`, `_.callback`, `_.chunk`, `_.clone`, `_.create`, `_.curry`, `_.curryRight`, `_.drop`, `_.dropRight`, `_.flatten`, `_.invert`, `_.max`, `_.min`, `_.parseInt`, `_.slice`, `_.sortBy`, `_.take`, `_.takeRight`, `_.template`, `_.trim`, `_.trimLeft`, `_.trimRight`, `_.trunc`, `_.random`, `_.range`, `_.sample`, `_.uniq`, & `_.words`
- Methods that work as an iteratee for `_.reduce`, & the like, out of the box:
  `_.assign`, `_.defaults`, `_.merge`, & `_.sortAllBy`

Other Changes

- Added cherry pickable method modules to the npm package
- Added `cancel` method to debounced & throttled functions
- Added `defaultValue` param to `_.result`
- Added `multiValue` param to `_.invert`
- Added `thisArg` param to `_.tap`
- Added `_.memoize.Cache` to enable `Map` / `WeakMap` to be used
- Added support for cloning array buffers & typed arrays to `_.clone` & `_.cloneDeep`
- Added support for comparing error objects & typed arrays to `_.isEqual`
- Enabled the `sourceURL` option of `_.template` to work in production builds
- Ensured `_.at` , `_.matches` , & `_.pullAt` work with falsey values when keys are given
- Ensured `_.callback` doesn’t error when `func` is nullish & `thisArg` is given
- Ensured `_.callback` supports binding built-in methods
- Ensured `_.debounce` & `_.throttle` work if the system time is set backwards
- Ensured `_.difference` works with arrays regardless of argument position
- Ensured `_.findWhere` & `_.where` always use `_.matches`
- Ensured `_.includes` supports `fromIndex` when iterating objects
- Ensured `_.indexOf([], undefined, true)` returns `-1` instead of `0`
- Ensured `_.intersection` ignores non-array secondary values
- Ensured `_.isEqual` works with wrapped objects containing `constructor` properties
- Ensured `_.keys` treats sparse arrays as dense
- Ensured `_.keys` works with string objects in IE < 9
- Ensured `_.matches` comparison isn’t affected by changes to `source` objects
- Ensured `_.max` & `_.min` return the correct value when `callback` computes ±Infinity
- Ensured `_.merge` ignores `undefined` values of `source` object properties
- Ensured `_.partial` & `_.partialRight` work with curried functions
- Ensured `_.pluck` always uses `_.property`
- Ensured `_.random` returns `1` or `0` when called with no arguments
- Ensured `_.range` , `_.slice` , & other methods handle `NaN` arguments
- Ensured `_.runInContext` uses a zeroed `_.uniqueId` counter
- Ensured `_.transform` checks that `object` is an object before using its `[[Prototype]]`
- Ensured `_.where` handles `source` objects with `undefined` property values
- Ensured `_.where` only returns elements that contain all source array/object values
- Ensured browserified lodash works in web workers
- Ensured customizing `_.indexOf` affects `_.includes` when iterating objects
- Ensured lodash works in `NW.js`
- Ensured lodash doesn’t add `Function.prototype` extensions to its prototype
- Fixed `_.isFunction` for typed array constructors in Safari 8
- Made `_.escape` & `_.unescape` handle backticks
- Made `_.isElement` more robust
- Made `_.parseInt` more closely follow spec
- Made `_.wrap` use `_.identity` when `wrapper` is nullish
- Made templates avoid referencing `_.escape` if “escape” delimiters are not used
- Made array-like object checks follow ES2015 `ToLength`
- Made use of `Set` in `_.difference` , `_.intersection` , & `_.uniq`
- Removed array & object pools
- Removed all method compilation from the `compat` build

- Updated the sourceURL syntax used by `_.template`

v2.4.2

***Apr. 26, 2015* — [Diff]() — [Docs]()**

- Avoided memory leaks in `_.bind`, `_.bindKey`, `_.curry`, `_.partial`, & `_.partialRight`

v2.4.1

***Dec. 2, 2013* — [Diff]() — [Docs]()**

- Ensured `__bindData__` is properly cloned
- Ensured `_.isEqual` can compare cyclical objects with shared property values
- Optimized `_.partial` & `_.partialRight`
- Reached ~100% code coverage

v2.4.0

***Nov. 25, 2013* — [Diff]() — [Docs]()**

- Added `_.constant`, `_.mapValues`, `_.now`, `_.property`, & `_.xor`
- Added an `options` param to `_.mixin` to specify whether functions added are chainable
- Added support for `_.sortBy` to accept an array of property names to sort by
- Enabled `_.zipObject` to accept an array of `keys` with no `values` param
- Removed conditional `setImmediate` use from `_.defer`

v2.3.0

***Nov. 10, 2013* — [Diff]() — [Docs]()**

- Added `_.create` & `_.noop`
- Avoided memory leaks in `_.debounce` & `_.throttle`
- Enhanced `_.createCallback` to avoid binding functions bound by `Function#bind`
- Ensured rebound functions correctly partially apply arguments
- Ensured `_.isEqual` works with values from `Object.create(null)` & `_(false)`
- Ensured `_.min` & `_.max` work as callbacks for `_.map`
- Ensured `_.template` coerces the `text` param to a string
- Optimized `_.difference`, `_.omit`, & `_.without` by way of `baseDifference`
- Optimized `_.isBoolean`, `_.isNumber`, & `_.isString` for the `false` case
- Optimized `_.sample` & `_.shuffle` by way of `baseRandom`
- Reduced `_.wrap` by way of `createBound`
- Removed native `Function#bind` use for better cross-environment consistency

v2.2.1

***Oct. 3, 2013* — [Diff]() — [Docs]()**

- Ensured `_.mixin` creates functions that work with `_.chain`
- Ensured the the `createObject` fallback is included in the `modern` build

v2.2.0

***Sept. 28, 2013* — [Diff]() — [Docs]()**

- Added `_.clone` support for date, regexp, & other built-in objects
- Ensured `_.random` avoids excessive results of `0` for floating-point numbers
- Made `compat` & `underscore` builds use `Date.now` when available
- Reduced dependencies on `getObject` & `releaseObject`

v2.1.0

***Sept. 22, 2013* — [Diff]() — [Docs]()**

- Added `Object.defineProperty` fallback for the `modern` build
- Added support to `_.random` to explicitly specify floating point numbers
- Enabled `_.compose` to be invoked without arguments
- Ensured `_.flatten` works with extremely large arrays
- Ensured `_.support` properties aren’t minified
- Ensured `reThis` isn’t used in Windows 8 applications
- Made UMD more resistant to false positives
- Optimized `_.isArguments` & `_.isArray` fallbacks

v2.0.0

***Sept. 13, 2013* — [Diff]() — [Docs]()**

Compatibility Warnings

- Aligned `_.after` with Underscore 1.5.0, making it always return a function
- Made `_.unzip` an alias of `_.zip`

Notable Changes

- Created lodash methods as npm packages & AMD/Node.js modules
- Made `_.chain` force chaining for all methods
- Moved the build utility to `lodash-cli`
- Optimized `_.contains`, `_.debounce`, `_.isArguments`, `_.throttle`, & `_.where`
- Optimized functions of `_.bind`, `_.bindKey`, `_.curry`, `_.partial`, & `_.partialRight`
- Added `_.curry`, `_.forEachRight`, `_.indexBy`, `_.findLast`, `_.findLastIndex`, `_.findLastKey`, `_.forInRight`, `_.forOwnRight`, `_.pull`, `_.remove`, & `_.sample`

Other Changes

- Added Curl & Dojo module loaders to the unit tests
- Added the `modularize` build option
- Added support for the `iife` command to be used without an `%output%` token
- Added support for `_.mixin` to accept a destination object
- Added support for `_.range` to accept a `step` of `0`
- Ensured “Array” methods support `arguments` objects
- Ensured “Function” methods throw on non-functions
- Ensured `_.at` works as a callback for `_.map`
- Ensured `_.createCallback` works when no `argCount` is specified
- Ensured `_.first` & `_.last` return arrays when given an `n` with a falsey `array`
- Ensured `_.flatten` works with `arguments` objects
- Ensured minified files work with Dojo’s builder
- Ensured `_.zipObject` skips falsey elements
- Improved dead code removal from builds

- Improved JSDoc syntax
- Made `_.eachRight` an alias of `_.forEachRight`
- Made `_.memoize` avoid prefixing `cache` keys when using a `resolver` function
- Removed local `clearTimeout` & `setTimeout` variables from the `underscore` build
- Reduced the size of the repo & npm package
- Simplified the bailout in `createCache`
- Updated sourceURL & sourceMappingURL syntax
- Updated `underscore` build compatibility to v1.5.2

v1.3.1

June 12, 2013* — *Diff* — *Docs

- Added missing `cache` property to the objects returned by `getObject`
- Ensured `maxWait` unit tests pass in Ringo
- Increased the `maxPoolSize` value
- Optimized `releaseArray` & `releaseObject`

v1.3.0

June 11, 2013* — *Diff* — *Docs

- Added `_.transform`
- Added `_.chain` & `_.findWhere` aliases
- Added internal array & object pooling
- Added Istanbul test coverage reports to Travis CI
- Added `maxWait` option to `_.debounce`
- Added support for floating point numbers to `_.random`
- Added Volo configuration to package.json
- Adjusted UMD for `component build`
- Enabled more stable mixing of `lodash` & `underscore` build methods
- Ensured debounced function with `leading` & `trailing` options works as expected
- Ensured minified builds work with the Dojo builder
- Ensured minification avoids deoptimizing expressions containing boolean values
- Ensured support for `--output` paths containing build command keywords
- Ensured unknown types return `false` in `_.isObject` & `_.isRegExp`
- Ensured `_.clone`, `_.flatten`, & `_.uniq` can be used as callback for `_.map`
- Ensured `_.forIn` works on objects with longer inheritance chains in IE < 9
- Ensured `_.isPlainObject` returns `true` for empty objects in IE < 9
- Ensured `_.max` & `_.min` chain correctly
- Ensured `clearTimeout` use doesn't cause errors in Titanium
- Ensured that the `--stdout` build option doesn't write to a file
- Exposed memoized function's `cache`
- Fixed `Error.prototype` iteration bugs
- Fixed "scripts" paths in component.json
- Made methods support customizing `_.indexOf`
- Made the build track dependencies of private functions
- Made the `template` pre-compiler build option avoid escaping non-ascii characters
- Made `_.createCallback` avoid binding functions if they don't reference `this`
- Optimized the Closure Compiler minification process

- Optimized the large array cache for `_.difference` , `_.intersection` , & `_.uniq`
- Optimized internal `_.flatten` & `_.indexOf` use
- Reduced `_.unzip` & `_.zip`
- Removed special handling of arrays in `_.assign` & `_.defaults`

v1.2.1

***Apr. 29, 2013* — [Diff]() — [Docs]()**

- Added Component package support
- Updated the build utility to work with changes in GitHub’s API
- Ensured `_.isPlainObject` works with objects created by `Object.create(null)`
- Ensured “isType” methods return `false` for subclassed values
- Ensured debounced functions, with `leading` & `trailing` calls enabled, only perform trailing calls after they’re called more than once

v1.2.0

***Apr. 16, 2013* — [Diff]() — [Docs]()**

- Added `_.unzip`
- Added an `options` param to `_.debounce` & `_.throttle`
- Enabled non- `underscore` builds to include `_.findWhere` & `_.chain`
- Ensured “Array” & “Object” methods work with `arguments` objects & arrays respectively
- Ensured build utility runs on Windows
- Ensured `underscore` build versions of “isType” methods align with Underscore
- Ensured methods avoid issues with the `__proto__` property
- Ensured `_.isEqual` uses a `callback` only if it’s a function
- Ensured `_.merge` applies a `callback` to nested properties
- Ensured `_.merge` provides the correct `callback` arguments when comparing objects
- Made lodash work with Browserify
- Removed all method compilation from the `modern` build

v1.1.1

***Mar. 27, 2013* — [Diff]() — [Docs]()**

- Ensured the `underscore` build of `_.forEach` accepts a `thisArg` param
- Updated vendor/tar to work with Node.js v0.10

v1.1.0

***Mar. 26, 2013* — [Diff]() — [Docs]()**

- Added `_.createCallback` , `_.findIndex` , `_.findKey` , `_.parseInt` , & `_.runInContext`
- Added `_.support`
- Added `callback` & `thisArg` params to `_.flatten`
- Added `rhino -require` support
- Added CommonJS/Node.js support to precompiled templates
- Ensured the `exports` object is not a DOM element
- Ensured `_.isPlainObject` returns `false` for objects without a `[[Class]]` of “Object”
- Made `callback` support in `_.cloneDeep` more closely follow its documentation

- Made `_.object` an alias of `_.zipObject`
- Made the template precompiler create missing directories of `--output` paths
- Optimized method chaining, object iteration, `_.find`, & `_.pluck`
- Updated `backbone` build method dependencies

v1.0.2

***Feb. 18, 2013* — [Diff]() — [Docs]()**

- Ensured `_.isPlainObject` works when built-in `Object.getPrototypeOf` is unavailable

v1.0.1

***Feb. 18, 2013* — [Diff]() — [Docs]()**

- Added support for specifying source map URLs in `-p` / `--source-map` options
- Ensured the second argument to `_.assign` is not treated as a `callback`
- Ensured `-p` / `--source-map` build options correctly set the `sourceMappingURL`
- Made `-p` / `--source-map` set source map “sources” keys based on the build
- Made `_.defer` use `setImmediate`, in Node.js, when available
- Made `_.where` search arrays for values regardless of their index position
- Removed dead code from `_.template`

v1.0.0

***Feb. 14, 2013* — [Diff]() — [Docs]()**

Compatibility Warnings

- Made `_.defaults` preserve `null` values, instead of overwriting them

Changes

- Added `_.at` & `_.partialRight`
- Added `modern` & `-p` / `--source-map` build options
- Added “imports” option to `_.templateSettings`
- Added support for `_.pluck` & `_.where` callback shorthands
- Ensured `_.assign` & `_.defaults` support arrays
- Ensured `_.merge` assigns `null` values & produces dense arrays
- Deferred minifier downloads until the `lodash` utility requires them
- Flipped `noNodeClass` test to avoid Firebug’s “break on all errors” feature
- Made `_.where` support deep object comparisons
- Optimized `_.invert`, `_.pairs`, & `_.values`
- Reduced `_.max`, `_.min`, `_.pluck`, `_.toArray`, & `_.where`
- Removed support for automatic with-statement removal from `_.template`
- Simplified `createIterator` & `iteratorTemplate`
- Tweaked `_.uniqueId` to avoid problems with buggy minifiers
- Updated `underscore` build compatibility to v1.4.4
- Added `callback` & `thisArg` params to `_.assign`, `_.clone`, `_.cloneDeep`, `_.first`, `_.last`, `_.initial`, `_.isEqual`, `_.merge`, & `_.rest`

v1.0.0-rc.3

***Dec. 17, 2012* — [Diff]() — [Docs]()**

Compatibility Warnings

- Made `_#join`, `_#pop`, & `_#shift` wrapper methods return unwrapped values
- Made “Function” methods wrapper counterparts return wrapped values
- Removed “chain” methods

Changes

- Added `_.cloneDeep`
- Added `_.once` to the `backbone` build
- Ensured `backbone` builds implement Underscore’s chaining behavior
- Ensured the `settings=…` build option doesn’t clobber the default `moduleId`
- Ensured lodash’s npm package works when packages aren’t globally installed
- Made compiled templates exported for AMD use the lodash module for `_`
- Removed the `_.forEach` dependency from `_.intersection`
- Optimized `_.isArray` & `_.isFunction` fallbacks as well as `_.intersection`, `_.isDate`, `_.isRegExp`, `_.reduce`, `_.reduceRight`, `_.union`, & `_.uniq`

v1.0.0-rc.2

***Dec. 5, 2012* — [Diff]() — [Docs]()**

- Specified more method chaining behaviors
- Updated `underscore` build compatibility to v1.4.3

v1.0.0-rc.1

***Dec. 4, 2012* — [Diff]() — [Docs]()**

Compatibility Warnings

- Added support for `arguments` objects to `_.clone`
- Ensured `_.clone` creates plain object clones of non-plain objects
- Made `_(…)` chain automatically without needing to call `_#chain`
- Made `_.isEqual` equate `arguments` objects to similar `Object` objects

Changes

- Ensure lodash runs in the JS engine embedded in Adobe products
- Ensured `_.reduce` & `_.reduceRight` provide the correct `callback` arguments
- Ensured `_.throttle` nulls the `timeoutId`
- Made deep `_.clone` more closely follow the structured clone algorithm
- Optimized compiled templates in Firefox
- Optimized `_.forEach`, `_.forOwn`, `_.isNumber`, & `_.isString`
- Simplified `iteratorTemplate`

v0.10.0

***Nov. 17, 2012* — [Diff]() — [Docs]()**

Compatibility Warnings

- Renamed `_.lateBind` to `_.bindKey`
- Made `_.defaults` & `_.extend` iterate only own properties of `source` objects to align with `Object.assign`

Changes

- Added the build commands used to custom build copyright/license headers
- Added `_.assign`
- Ensured the `underscore` build of `_.find` returns the first match, not last
- Ensured `_.defaults`, `_.extends`, & `_.merge` work with `_.reduce`
- Made lodash’s npm package installation work with more system configurations
- Made `_.extend` an alias of `_.assign`
- Optimized `_.contains`, `_.defaults`, `_.extend`, & `_.filter`
- Restricted `_.where` to iterate only own properties of `source` objects
- Updated `backbone` build lodash method dependencies

v0.9.2

***Nov. 9, 2012* — [Diff]() — [Docs]()**

- Added `fromIndex` param to `_.contains`
- Added `moduleId` build option
- Added Closure Compiler “simple” optimizations to the build process
- Added support for strings in `_.max` & `_.min`
- Added support for ES2015 template delimiters to `_.template`
- Ensured re-minification of lodash by third parties avoids Closure Compiler bugs
- Optimized `_.every`, `_.find`, `_.some`, & `_.uniq`

v0.9.1

***Oct. 31, 2012* — [Diff]() — [Docs]()**

- Ensured `_.every` returns `false` as soon as the `callback` result is falsey
- Ensured `_.isFinite` returns `false` for non-numeric values
- Removed `_.forEach` chainability in the `underscore` build
- Simplified `_.union`

v0.9.0

***Oct. 24, 2012* — [Diff]() — [Docs]()**

- Added a `sourceURL` option to `_.template`
- Ensured `_.where` returns an empty array if given an empty `properties` object
- Expanded `_.isFinite` to return `true` for numeric strings
- Reduced `_.intersection`, `_.omit`, `_.pick`, `_.sortedIndex`, & `_.where`
- Reduced the npm package size by only downloading minifiers for global installs
- Reduced lodash’s file size
- Improved source code comprehension by removing compilation from `_.bindAll`, `_.contains`, `_.countBy`, `_.every`, `_.filter`, `_.find`, `_.functions`, `_.groupBy`, `_.invert`, `_.invoke`, `_.isEmpty`, `_.map`, `_.merge`, `_.omit`, `_.pairs`, `_.pick`, `_.pluck`, `_.reduce`, `_.reject`, `_.some`, `_.sortBy`, `_.values`, `_.where`, & internal `shimKeys`

v0.8.2

***Oct. 10, 2012* — [Diff]() — [Docs]()**

- Ensured `_.map` returns an array when given a falsey collection
- Ensured `_.throttle` clears its timeout when `func` is called

- Made `_.max`, `_.min`, `_.shuffle` support iterating objects
- Reduced `createIterator`, `_.clone`, `_.compact`
- Re-optimized `_.max`, `_.min`, & `_.sortedIndex`

v0.8.1

***Oct. 4, 2012* — Diff — Docs**

- Enabled deep clone support in the `underscore` build when `clone` is explicitly included
- Reverted removal of first argument falsey checks from methods

v0.8.0

***Oct. 1, 2012* — Diff — Docs**

Compatibility Warnings

- Made `_.random` return `0` or `1` when no arguments are given
- Moved late bind functionality from `_.bind` to `_.lateBind`
- Removed first argument falsey checks from methods
- Removed support for custom `clone`, `isEqual`, `toArray` methods from `_.clone`, `_.isEqual`, & `_.toArray`

Changes

- Added `-d` / `--debug`, `-m/--minify`, `minus`, `plus`, `settings`, & `template` build options
- Added `_.isPlainObject` & `_.lateBind`
- Enabled `_.sortedIndex` to accept a property name as the `callback` param
- Ensured methods accept a `thisArg` of `null`
- Fixed the `iife` build option to accept more values
- Made `_.times` return an array of `callback` results
- Simplified `_.max`, `_.min`, & `_.reduceRight`

v0.7.0

***Sept. 11, 2012* — Diff — Docs**

Compatibility Warnings

- Renamed `_.zipObject` to `_.object`
- Replaced `_.drop` with `_.omit`
- Made `_.drop` an alias of `_.rest`

Changes

- Added `_.invert`, `_.pairs`, & `_.random`
- Added `_.result` to the `backbone` build
- Added `exports`, `iife`, `-c` / `--stdout`, `-o` / `--output`, & `-s` / `--silent` build options
- Ensured `isPlainObject` works with objects from other documents
- Ensured `_.isEqual` compares values with circular references correctly
- Ensured `_.merge` work with four or more arguments
- Ensured `_.sortBy` performs a stable sort for `undefined` values
- Ensured `_.template` works with “interpolate” delimiters containing ternary operators
- Ensured the production build works in Node.js
- Ensured template delimiters are tokenized correctly
- Made pseudo private properties `_chain` & `_wrapped` double-underscored to avoid conflicts

- Made `minify.js` support `underscore.js`
- Reduced the size of `mobile` & `underscore` builds
- Simplified `_.isEqual` & `_.size`

v0.6.1

***Aug. 29, 2012* — [Diff]() — [Docs]()**

- Ensured IE conditional compilation isn't enabled by the `useSourceURL` test
- Optimized `isPlainObject`

v0.6.0

***Aug. 28, 2012* — [Diff]() — [Docs]()**

- Added `callback` & `thisArg` params to `_.drop` & `_.pick`
- Added `hasObjectSpliceBug` test to avoid `delete` operator use
- Added `_.unescape`
- Ensured `_.reduce` works with string objects in IE < 9
- Made `_.omit` an alias of `_.drop`
- Made compiled methods take advantage of engines with strict mode optimizations
- Optimized `_.intersection` & removed its dependency on `_.every`
- Reduced the file size of the `underscore` build

v0.5.2

***Aug. 21, 2012* — [Diff]() — [Docs]()**

- Ensured `_.isElement` uses strict equality for its duck type check
- Ensured `_.isObject` returns a boolean value
- Ensured `_.template` & "Object" methods don't error when given falsey values
- Made `_.template` generate less unused code in compiled templates

v0.5.1

***Aug. 18, 2012* — [Diff]() — [Docs]()**

- Ensured `_.bind` correctly appends array arguments to partially applied arguments in older browsers

v0.5.0

***Aug. 17, 2012* — [Diff]() — [Docs]()**

- Added `_.countBy`, `_.drop`, `_.merge`, & `_.where`
- Added `csp` *(Content Security Policy)* & `underscore` build options
- Added deep cloning support to `_.clone`
- Added Jam package support
- Added support for exiting early in `_.forEach`, `_.forIn`, & `_.forOwn`
- Added support for jQuery/MooTools DOM query collections to `_.isEmpty` & `_.size`
- Ensured development build works with IE conditional compilation enabled
- Ensured `_.clone` doesn't clone `arguments` objects, DOM nodes, functions, or non-plain objects
- Ensured `_.filter`'s `callback` can't modify result values
- Ensured `_.isEmpty`, `_.isEquals`, & `_.size` support `arguments` objects

- Ensured `_.isEqual` doesn't inspect DOM nodes, works with objects from other documents, & calls custom `isEqual` methods before checking strict equality
- Ensured `_.once` frees the given function for garbage collection
- Ensured `_.sortBy` performs a stable sort
- Ensured `reEvaluateDelimiter` is used when `_.templateSettings.evaluate` is `undefined`
- Made `_.range` coerce arguments to numbers
- Optimized `_.isFunction`

v0.4.2

***July 16, 2012* — [Diff]() — [Docs]()**

- Added `strict` build option
- Ensured `_.bindAll`, `_.defaults`, & `_.extend` avoid strict mode errors on read-only properties
- Optimized the iteration of large arrays in `_.difference`, `_.intersection`, & `_.without`
- Fixed build bugs related to removing variables

v0.4.1

***July 11, 2012* — [Diff]() — [Docs]()**

- Fixed `_.template` regression
- Optimized build process to detect & remove more unused variables

v0.4.0

***July 11, 2012* — [Diff]() — [Docs]()**

- Added `bin` & `scripts` entries to package.json
- Added `legacy` build option
- Added cross-browser support for strings to "Collection" methods
- Added `_.zipObject`
- Leveraged `_.indexOf`'s `fromIndex` in `_.difference` & `_.without`
- Optimized compiled templates
- Optimized inlining the `iteratorTemplate` for builds
- Optimized object iteration for "Collection" methods
- Optimized partially applied `_.bind` in V8
- Made compiled templates more debuggable
- Made `_.size` work with falsey values & consistent cross-browser with `arguments` objects
- Moved `_.groupBy` & `_.sortBy` back to the "Collection" category
- Removed `arguments` object from `_.range`

v0.3.2

***June 14, 2012* — [Diff]() — [Docs]()**

- Ensured `_.escape` returns an empty string for nullish values
- Ensured `sourceURL` support doesn't cause errors in Adobe's JS engine
- Fixed regression in generating custom builds
- Moved `_.invoke` & `_.pluck` back to the "Collection" category
- Moved `_.tap` to the "Seq" category

v0.3.1

***June 10, 2012* — [Diff] — [Docs]**

- Added `backbone` build option
- Ensured “Array” methods allow falsey `array` params
- Removed `_.isArguments` fallback from the `mobile` build
- Simplified `_.pluck`, `_.values` & `_(...)` method wrappers

v0.3.0

***June 6, 2012* — [Diff] — [Docs]**

- Added `_.forIn` & `_.forOwn`
- Added `category` build option
- Added `fromIndex` param to `_.indexOf` & `_.lastIndexOf`
- Added `//@ sourceURL` support to `_.template`
- Added `thisArg` param to `_.sortedIndex` & `_.uniq`
- Ensured array-like objects with invalid `length` properties are treated like regular objects
- Ensured `_.sortedIndex` supports arrays with high `length` values
- Fixed `prototype` iteration bug in `_.keys`
- Optimized `_.times` & `this` bindings in iterator methods

v0.2.2

***May 30, 2012* — [Diff] — [Docs]**

- Added `mobile` build option
- Ensured `_.find` returns `undefined` for unmatched values
- Ensured `_.templateSettings.variable` is compatible with Underscore
- Optimized `_.escape`
- Reduced dependencies of `_.find`

v0.2.1

***May 24, 2012* — [Diff] — [Docs]**

- Adjusted the lodash export order for r.js
- Ensured `_.groupBy` values are added to own, not inherited, properties
- Made `_.bind` follow ES5 spec to support a popular Backbone.js pattern
- Removed the alias `_.intersect`
- Simplified `_.bind`, `_.flatten`, `_.groupBy`, `_.max`, & `_.min`

v0.2.0

***May 21, 2012* — [Diff] — [Docs]**

- Added `_.partial`
- Added `thisArg` param to `_.groupBy`
- Added “lazy bind” support to `_.bind`
- Added custom build options
- Added default `_.templateSettings.variable` value

- Added native method overwrite detection to avoid bad native shims
- Added support for more AMD build optimizers & aliasing as the “underscore” module
- Added whitespace to compiled strings
- Commented the `iterationFactory` options object
- Ensured `_(...)` returns given wrapper instances
- Ensured `_.max` & `_.min` support extremely large arrays
- Ensured `_.throttle` works in tight loops
- Fixed `clearTimeout` typo
- Fixed `[DontEnum]` bug in IE < 9
- Fixed `prototype` iteration bug in Firefox < 3.6, Opera < 11.60, & Safari < 5.1
- Inlined `_.isFunction` calls
- Made `_.debounce`’ed functions match `_.throttle`’ed functions’ return value behavior
- Made `_.escape` no longer translate the “>” character
- Simplified all “Array” methods
- Optimized `_.debounce`, `_.escape`, `_.flatten`, `_.forEach`, `_.groupBy`, `_.intersection`, `_.invoke`, `_.isObject`, `_.max`, `_.min`, `_.pick`, `_.shuffle`, `_.sortedIndex`, `_.template`, `_.throttle`, `_.union`, `_.uniq`

v0.1.0

***Apr. 23, 2012* — [Docs]()**

- Initial release