J.P. Strydom

What Is Lodash?

Why Use Lodash?

Some Usefu Lodash

Function

- _.map
- _.set
- _.assign
- _.debounce
- _.find
- _.deburr
- lrow Day
- _.cloneDe

Questions?



Lodash

What is Lodash & why should I use it?

J.P. Strydom

Entelect

Cell C

March 26, 2018

Outline

- _.map
- _.get

- _.debounce
- _.deburr

2 Why Use Lodash?

What Is Lodash?

- Some Useful Lodash Functions
 - _.map
 - _.get
 - _.set
 - _.assign
 - _.debounce
 - _.find
 - _.deburr
 - _.keyBy
 - _.cloneDeep

J.P. Strydom

What Is Lodash?

Why Use Lodash?

Some Usefu

Lodash

Function

- _.map
- _.get
- _.set
- _.assign
- _.debounce
- _.find
- _.deburr
- -.depuil
- ...cloneDe

Questions?

What Is Lodash?

J.P. Strydom

What Is Lodash?

_.map

- _.get

- _.debounce
- _.deburr

What Is Lodash?

Lodash, or "_" for short, is a JavaScript helper library for arrays, strings, and objects. It provides a ton of helper functions to help you save time.

Documentation here: https://lodash.com/docs

J.P. Strydom

What Is Lodash?

Why Use Lodash?

Some Usetu Lodash

Function

- _.map
- _.get
- _.set
- _.assign
- _.debounce
- _.find
- _.deburr
- -. debuil
- cloneDe

J.P. Strydom

Why Use Lodash?

_.map

_.get _.set

_.assign

_.debounce

_.deburr

Why Use Lodash?

• It has over 400 useful helper functions.

J.P. Strydom

Why Use Lodash?

- _.map
- _.get
- _.debounce
- _.deburr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.

J.P. Strydom

Why Use Lodash?

- _.map _.get

- _.debounce
- _.deburr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.

J.P. Strydom

Why Use Lodash?

- _.map
- _.get

- _.debounce
- _.deburr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.
 - arrays
 - strings
 - objects

J.P. Strydom

Why Use Lodash?

- _.map
- _.assign
- _.debounce
- _.deburr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.
 - arrays
 - strings
 - objects
- It's well-documented, well-supported, and well-loved.

J.P. Strydom

Why Use Lodash?

- _.map
- _.assign
- debounce
- _.deburr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.
 - arrays
 - strings
 - objects
- It's well-documented, well-supported, and well-loved.
 - 0 issues on GitHub.

J.P. Strydom

Why Use Lodash?

- _.map
- _.assign
- debounce
- _.deburr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.
 - arrays
 - strings
 - objects
- It's well-documented, well-supported, and well-loved.
 - 0 issues on GitHub.
 - 30,000+ stars on GitHub.

J.P. Strvdom

Why Use Lodash?

- _.map
- _.assign
- debounce
- _.deburr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.
 - arrays
 - strings
 - objects
- It's well-documented, well-supported, and well-loved.
 - 0 issues on GitHub.
 - 30,000+ stars on GitHub.
 - 64,000+ dependents on NPM.

J.P. Strvdom

Why Use Lodash?

- _.map
- _.assign

- dehurr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.
 - arrays
 - strings
 - objects
- It's well-documented, well-supported, and well-loved.
 - 0 issues on GitHub.
 - 30,000+ stars on GitHub.
 - 64,000+ dependents on NPM.
- It's thoroughly tested across multiple platforms and browsers.

J.P. Strydom

What Is Lodash?

Why Use Lodash?

Some Useful Lodash

- Functions
- _.map
- _.set
- _.assign
- _.debounce
- find
- _.deburr
- _.keyBy
- _.CIOHebee

Questions

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.
 - arrays
 - strings
 - objects
- It's well-documented, well-supported, and well-loved.
 - 0 issues on GitHub.
 - 30,000+ stars on GitHub.
 - 64,000+ dependants on NPM.
- It's thoroughly tested across multiple platforms and browsers.
- It's very efficient (more efficient than native JS methods).

J.P. Strvdom

Why Use Lodash?

- _.map
- _.assign

- dehurr

- It has over 400 useful helper functions.
- It leads to clean, easy-to-use, and easy-to-read code.
- It automatically handles multiple variable types for you.
 - arrays
 - strings
 - objects
- It's well-documented, well-supported, and well-loved.
 - 0 issues on GitHub.
 - 30,000+ stars on GitHub.
 - 64,000+ dependents on NPM.
- It's thoroughly tested across multiple platforms and browsers.
- It's very efficient (more efficient than native JS methods).
- It's very light (69 KB)

J.P. Strydom

What Is

Why Use Lodash?

Some Useful Lodash

Functions

- _.map
- _.get
- _.set
- _.assign
- _.debounce
- _.find
- _.deburr
- _.keyBy
- _.CIONEDE

Questions

Some Useful Lodash Functions

Let's look at a few useful Lodash methods...

Lodash?

Lodash Functions

- ..map
- _.set
- _.assign
- _.debounce
- 4:-3
- _.deburr
- _.keyBy
- _.cloneDe

Questions?

_.map(collection, iteratee)

Creates an array of values by running each element in collection through iteratee.

- _.map _.get
- _.assign
- debounce
- _.deburr

_.map(collection, iteratee)

Creates an array of values by running each element in collection through iteratee.

collection (Array | Object): The collection to iterate over. iteratee (Function): The function invoked per iteration.

The iteratee is invoked with three arguments: (value, index|key, collection).

- _.map
- _.get
- _.assign

- deburr

_.map(collection, iteratee)

Creates an array of values by running each element in collection through iteratee.

collection (Array | Object): The collection to iterate over. iteratee (Function): The function invoked per iteration.

The iteratee is invoked with three arguments: (value, index|key, collection).

returns (Array): The new mapped array.

```
Lodash
```

J.P. Strydom

 $_{-}.\mathtt{map}$

```
_.map
```

_.get

_.set _.assign

dehounce

dehurr

Example

```
function square(n) {
        return n * n;
3
4
5
      _.map([4, 8], square);
6
      // => [16, 64]
8
      _.map({ 'a': 4, 'b': 8 }, square);
9
      // => [16, 64] (iteration order is not guaranteed)
10
11
      const users = [
12
        { 'user': 'barney' },
13
        { 'user': 'fred' }
14
      ];
15
16
      // Using the '_.property' iteratee shorthand.
17
      _.map(users, 'user');
18
      // => ['barney', 'fred']
```

Some Usefu

Lodash Functions

- _.map
- _.set
- _.assign
- _.debounce
- find
- _.deburr
- cloneDee

Questions?

_.get(object, path, [defaultValue])

Gets the value at path of object. If the resolved value is undefined, the defaultValue is returned in its place.

Some Usefu

Lodash
Functions

- _.map
- _.set
- _.assign
- _.debounce
- find
- _.deburr
- _.keyBy
- _.clouenee

Questions?

_.get(object, path, [defaultValue])

Gets the value at path of object. If the resolved value is undefined, the defaultValue is returned in its place.

object (Array | Object): The object to query.

path (Array | string): The path of the property to get.

defaultValue (*): The value returned for undefined results.

Some Useful

Lodash Functions

- _.map
- _.get _.set
- _.assign
- _.debounce
- find
- _.deburr
- _.keyBy
- _.cloneDee

Questions

```
_.get(object, path, [defaultValue])
```

Gets the value at path of object. If the resolved value is undefined, the defaultValue is returned in its place.

object (Array | Object): The object to query.
path (Array | string): The path of the property to get.
defaultValue (*): The value returned for undefined results.

returns (*): The resolved value.

```
Lodash
```

J.P. Strydom

_.get

```
_.map
_.get
```

- _.set
- _.assign
- _.debounce
- _.deburr

Example

```
const object = { 'a': [{ 'b': { 'c': 3 } }] };
2
      _.get(object, 'a[0].b.c');
4
      // => 3
5
6
      _.get(object, ['a', '0', 'b', 'c']);
7
      // => 3
8
9
      _.get(object, 'a.b.c', 'default');
10
      // => 'default'
```

- _.map
- _.get _.set
- _.assign
- dehounce
- _.deburr

_.set(object, path, value)

Sets the value at path of object. If a portion of path doesn't exist, it's created. Arrays are created for missing index properties while objects are created for all other missing properties.

- _.map
- _.get _.set
- _.assign
- dehounce
- _.deburr

_.set(object, path, value)

Sets the value at path of object. If a portion of path doesn't exist, it's created. Arrays are created for missing index properties while objects are created for all other missing properties.

object (Array | Object): The object to modify. path (Array | string): The path of the property to set. value (*): The value to set.

Some Useful

Lodash Functions

- _.map
- _.set
- _.assign
- _.debounce
- _.find
- _.deburr
- _.cloneDee

Questions

_.set(object, path, value)

Sets the value at path of object. If a portion of path doesn't exist, it's created. Arrays are created for missing index properties while objects are created for all other missing properties.

object (Array | Object): The object to modify.

path (Array | string): The path of the property to set.

value (*): The value to set.

returns (Object): The mutated object.

```
Lodash
```

J.P. Strydom

_.set

Lodash?

Some Usefi Lodash

Lodash Functions

..map ..get

_.set

_.debounce

_.debounce

_.deburr

.keyBy

Questions?

Example

```
const object = { 'a': [{ 'b': { 'c': 3 } }] };

..set(object, 'a[0].b.c', 4);
console.log(object.a[0].b.c);

// => 4

..set(object, ['x', '0', 'y', 'z'], 5);
console.log(object.x[0].y.z);

// => 5
```

- _.map _.get
- _.assign
- debounce
- _.deburr

_.assign(object, [sources])

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

Some Usefu

Lodash
Functions

_.map _.get

_.set

_.assign

_.debound

_.deburr

.kevB

_.cloneDe

Questions

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

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

object (Object): The destination object. sources (...Object): The source objects.

- _.map _.get
- _.assign
- debounce
- _.deburr

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

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

object (Object): The destination object. sources (... Object): The source objects.

returns (Object): The mutated object.

```
Lodash
```

J.P. Strydom

_.assign

Lodash?

Lodash?

Lodash

Functions

_.map

_.get

_.assign

_.debounce

...find

..deburr

deburr keyBy

et ot 1

Example

```
1   const a = {'a': 0};
2
3   const b = {'b': 1};
4
5   _.assign({ 'a': 1 }, a, b);
6   // => { 'a': 0, 'b': 1 }
```

_.map _.get

_.assign

_.debounce

_.deburr

_.debounce(func, [wait], [options])

Creates a debounced function that delays invoking func until after wait milliseconds have elapsed since the last time the debounced function was invoked.

- _.map _.get
- _.assign
- _.debounce
- _.deburr

_.debounce(func, [wait], [options])

Creates a debounced function that delays invoking func until after wait milliseconds have elapsed since the last time the debounced function was invoked.

func (Function): The function to debounce.

wait (number): The number of milliseconds to delay.

options (Object): The options object.

- _.map _.get
- _.assign
- _.debounce

- _.deburr

_.debounce(func, [wait], [options])

Creates a debounced function that delays invoking func until after wait milliseconds have elapsed since the last time the debounced function was invoked.

func (Function): The function to debounce.

wait (number): The number of milliseconds to delay.

options (Object): The options object.

returns (Function): The new debounced function.

```
Lodash
```

_.debounce

Lodash?

Lodash?

Some Useful Lodash

_.map

_.set

_.debounce

_.find

_.deburr

_.cloneDe

Questions?

```
function search(someSearchString) {
2
           // Preform a search API call
3
      }
 4
5
      const searchValue = document.getElementById(
6
           "search-field"
      );
8
      searchValue.addEventListener(
9
           "keyup",
10
          _.debounce(search, 500)
11
      );
```

 $_{-}$.find

What Is Lodash?

_.lind(collection

Some Useful Lodash Functions

- _.map _.get
- _.set
- _.assign
- _.debounce
- _.find
- _.deburr
- _.cloneDe

Questions?

_.find(collection, [predicate], [fromIndex])

Iterates over elements of collection, returning the first element predicate returns truthy for.

- _.map
- _.get
- _.assign
- debounce
- _.find _.deburr

_.find(collection, [predicate], [fromIndex])

Iterates over elements of collection, returning the first element predicate returns truthy for.

collection (Array | Object): The collection to inspect. iteratee (Function): The function invoked per iteration.

fromIndex (number): The index to search from.

The predicate is invoked with three arguments: (value, index | key, collection).

What Is Lodash?

Some Useful

Lodash
Eunctions

- _.map
- _.get
- _.assign
- dehounce
- _.debounc
- ..find
- _.aebu
- _.cloneDe

Questions

```
_.find(collection, [predicate], [fromIndex])
```

Iterates over elements of collection, returning the first element predicate returns truthy for.

collection (Array | Object): The collection to inspect. iteratee (Function): The function invoked per iteration. fromIndex (number): The index to search from.

The predicate is invoked with three arguments: (value, index | key, collection).

returns (*): The matched element, else undefined.

What Is Lodash?

Why Use Lodash?

Some Useful Lodash Functions

_.map _.get

_.set

_.debounce

..deburr

_.keyBy _.cloneDee

_.CIOHebee

```
1
      const users = [
        { 'user': 'barney', 'age': 36, 'active': true },
3
        { 'user': 'fred', 'age': 40, 'active': false },
4
        { 'user': 'pebbles', 'age': 1, 'active': true }
5
      ];
6
      _.find(users, o => o.age < 40);
8
      // => object for 'barney'
9
      // The '.matches' iteratee shorthand.
10
11
      _.find(users, { 'age': 1, 'active': true });
12
      // => object for 'pebbles'
13
14
      // The '_.matchesProperty' iteratee shorthand.
15
      _.find(users, ['active', false]);
16
      // => object for 'fred'
17
18
      // The '_.property' iteratee shorthand.
19
      _.find(users, 'active');
20
      // => object for 'barney'
```

_.deburr

What Is Lodash?

Some Usef Lodash

Lodash Functions

- _.map
- _.set
- _.assign
- _.debounce
- ..find
- _.deburr
- _.cloneDe

Questions

_.deburr([string])

Deburrs string by converting Latin-1 Supplement and Latin Extended-A letters to basic Latin letters and removing combining diacritical marks.

_.deburr

- _.map _.get
- _.assign
- _.debounce
- _.deburr

_.deburr([string])

Deburrs string by converting Latin-1 Supplement and Latin Extended-A letters to basic Latin letters and removing combining diacritical marks.

string (*string*): The string to deburr.

_.deburr

_.map

_.get

_.assign

debounce

_.deburr

_.deburr([string])

Deburrs string by converting Latin-1 Supplement and Latin Extended-A letters to basic Latin letters and removing combining diacritical marks.

string (string): The string to deburr.

returns (string): The deburred string.

```
Lodash
```

_.deburr

Lodash?

Some Usefi

Lodash

Function

_.map

- _.get
- _.set
- _.assign
- _.debounce
- _.debounc
- _.find
- _.deburr
- . kevBv
- _.cloneDee

```
1 _.deburr('déjà vu');
2 // => 'deja vu'
```

_.map

_.get

_.assign

debounce

_.deburr

_.keyBy

_.keyBy(collection, [iteratee])

Creates an object composed of keys generated from the results of running each element of collection through iteratee. The corresponding value of each key is the last element responsible for generating the key.

- _.map
- _.get
- _.assign

- _.deburr _.keyBy

_.keyBy(collection, [iteratee])

Creates an object composed of keys generated from the results of running each element of collection through iteratee. The corresponding value of each key is the last element responsible for generating the key.

collection (Array | Object): The collection to iterate over. iteratee (Function): The iteratee to transform keys.

The iteratee is invoked with one argument: (value).

- _.map
- _.get
- _.assign

- _.deburr _.keyBy

_.keyBy(collection, [iteratee])

Creates an object composed of keys generated from the results of running each element of collection through iteratee. The corresponding value of each key is the last element responsible for generating the key.

collection (Array | Object): The collection to iterate over. iteratee (Function): The iteratee to transform keys.

The iteratee is invoked with one argument: (value).

returns (Object): The composed aggregate object.

What Is Lodash?

Why Use Lodash?

Some Usefu Lodash

_.map

_.set _.assign

..find

_.deburr _.keyBy

_.cloneDee

Questions?

```
1
      const array = [
        { 'dir': 'left', 'code': 97 }.
3
        { 'dir': 'right', 'code': 100 }
4
      ];
5
6
      _.keyBy(array, o => String.fromCharCode(o.code));
      /*
8
      => {
9
           'a': { 'dir': 'left', 'code': 97 },
          'd': { 'dir': 'right', 'code': 100 }
10
11
12
      */
13
14
      _.keyBy(array, 'dir');
15
      /*
16
      => {
17
           'left': { 'dir': 'left', 'code': 97 },
           'right': { 'dir': 'right', 'code': 100 }
18
19
      */
20
```

_.map

_.get

_.set _.assign

_.debounce

_.deburr

_.cloneDeep

_.cloneDeep(value)

Recursively deeply clones value.

- _.map _.get

- _.debounce
- _.deburr
- _.cloneDeep

_.cloneDeep(value)

Recursively deeply clones value.

value (*): The value to recursively clone.

What Is Lodash?

Some Usefu

Lodash Functions

_.map

_.get

_.assign

_.debounce

..find

_.deburr

_.ReyBy
_.cloneDeep

_.CIONEDE

Questions?

_.cloneDeep(value)

Recursively deeply clones value.

value (*): The value to recursively clone.

returns (*): The deep cloned value.

```
Lodash
```

_.cloneDeep

- _.map
- _.get _.set
- _.assign
- _.debounce

- _.deburr
- _.cloneDeep

```
const objects = [{ 'a': 1 }, { 'b': 2 }];
2
3
     const deep = _.cloneDeep(objects);
     console.log(deep[0] === objects[0]);
4
5
     // => false
```

Lodash

J.P. Strydom

What Is Lodash?

Lodash?

Some Usefu Lodash

Function

_.map

_.get

..set

_.debounce

_.debound

..find

lrow Day

_.cloneDe

Questions?

Questions?

