# README

The functional.js library is an alternative to libraries such as underscore.js and lodash.js. It is of very minimal size. It translates modern javascript methods into their functional representation and maps them to the global _ object.

***The following Array prototype functions are mapped to the _ object:***

1. **every**

```
> _.every(that = [1,2,3], function(){
  return true;
})
```
*true*
```
> that
```
*[1,2,3]*

2. **some**

```
> _.some(that = [1,2,3], function(x){
  return x === 2;
})
```
*true*
```
> that
```
*[1,2,3]*

3. **filter**

```
> _.filter(that = [1,2,3], function(){
  return true;
})
```
*[1,2,3]*
```
> that
```
*[1,2,3]*

4. **map**

```
> _.map(that = [1,2,3], function(x){
  return 2*x;
})
```
*[2,4,6]*
```
> that
```
*[1,2,3]*

5. **reduce**

```
> _.reduce(that = [1,2,3], function(memo, x){
  return memo/x;
})
```

*0.16666666666666666*

```
> that
```

*[1,2,3]*

6. **reduceRight**

```
> _.reduceRight(that = [1,2,3], function(memo, x){
  return memo/x;
})
```

*1.5*

```
> that
```

*[1,2,3]*

7. **forEach**

```
> _.forEach(that = [1,2,3], function(x){
  that = x;
})
```

*undefined*

```
> that
```

*3*

8. **concat**

```
> _.concat(that = [1,2,3], 4)
```

*[1,2,3,4]*

```
> that
```

*[1,2,3]*

9. **join**

```
> _.join(that = [1,2,3], "-")
```

*"1-2-3"*

```
> that
```

*[1,2,3]*

10. **indexOf**

```
> _.indexOf(that = [1,2,2,3], 2)
```

*1*

```
> that
```

*[1,2,2,3]*

11. **lastIndexOf**

```
> _.lastIndexOf(that = [1,2,2,3], 2)
```

*2*

```
> that
```

*[1,2,2,3]*

12. **reverse**

```
> _.reverse(that = [1,2,3])
```
*[3,2,1]*
```
> that
```
*[3,2,1]*

13. **pop**

```
> _.pop(that = [1,2,3])
```
*3*
```
> that
```
*[1,2]*

14. **shift**

```
> _.shift(that = [1,2,3])
```
*1*
```
> that
```
*[2,3]*

15. **sort**

```
> _.sort(that = [3,2,1])
```
*[1,2,3]*
```
> that
```
*[1,2,3]*

16. **slice**

```
> _.slice(that = [1,2,3],1)
```
*[2,3]*
```
> that
```
*[1,2,3]*

***The follow Array prototype functions are also mapped to the _ object. However they return the first argument passed instead of the result of the mapped function:***

1. **push**

```
> _.push(that = [1,2,3], 4)
```
*[1,2,3,4]*
```
> that
```
*[1,2,3,4]*

2. **unshift**

```
> _.unshift(that = [1,2,3], 4)
```
*[4,1,2,3]*

```
> that
```
*[4,1,2,3]*

3. **splice**

```
> _.splice(that = [1,2,3,4,5,6], 2, 2, "a", "b")
```
*[1,2,"a","b",5,6]*
```
> that
```
*[1,2,"a","b",5,6]*

***The following Object functions are mapped to the _ object:***

1. **create**

```
> _.create(that = { a: 1 })
```
*{}*
```
> that
```
*{"a":1}*

2. **is**

```
> _.is(that = { a: 1 }, that)
```
*true*
```
> that
```
*{"a":1}*

3. **keys**

```
> _.keys(that = { a: 1, b: 2 })
```
*["a","b"]*
```
> that
```
*{"a":1,"b":2}*

***The following Object prototype function is mapped to the _ object:***

1. **hasOwnPrototype**

Mapped to "has".

```
> _.has(that = { a: 1 }, "a")
```
*true*
```
> that
```
*{"a":1}*

***The following Function prototype function is mapped to the _ object:***

1. **bind**

```
> _.bind(function(){
  return this;
}, { a: 1 })()
{"a":1}
```

***The following Array object function is mapped to the _ object:***

1. **isArray**

```
> _.isArray(that = { 0: 1, length: 1 })
false
> that
{"0":1,"length":1}
```

***The following are custom functions added to the _ object:***

1. **each**

   Converts the first argument to an array, if not already. It then invokes the method forEach over it with the remaining parameters. Returns the first argument.

   If the first argument is a pseudo array then it is converted to an array of values, otherwise it is converted to an array of keys.

```
> _.each(that = { 0: 1, 1: 2, 2: 3, length: 3 }, function(x){
  that = x;
})
{"0":1,"1":2,"2":3,"length":3}
> that
3
> _.each(that = { a: 1, b: 2, c: 3 }, function(x){
  that = x;
})
{"a":1,"b":2,"c":3}
> that
"c"
```

2. **toArray**

   Returns the first argument after converting it to an array, if not already.

   If the first argument is a pseudo array then it is converted to an array of values, otherwise it is converted to an array of keys.

```
> _.toArray(that = { 0: 1, 1: 2, 2: 3, length: 3 })
[1,2,3]
> that
{"0":1,"1":2,"2":3,"length":3}
> _.toArray(that = { a: 1, b: 2, c: 3 })
```

*["a","b","c"]*

```
> that
```

*{"a":1,"b":2,"c":3}*

3. **extend**

```
> _.extend(that = { a: 1 }, { a: 2, b: 3 }, { c: 4 })
```

*{"a":2,"b":3,"c":4}*

```
> that
```

*{"a":2,"b":3,"c":4}*

4. **clone**

```
> _.clone(that = { a: 2, b: 3 }, { c: 4 })
```

*{"a":2,"b":3,"c":4}*

```
> that
```

*{"a":2,"b":3}*

5. **size**

```
> _.size(that = { a: 2, b: 3 })
```

*2*

```
> that
```

*{"a":2,"b":3}*

```
> _.size(that = { length: 4 })
```

*4*

```
> that
```

*{"length":4}*

6. **call**

```
> _.call(function(x){
  return 2 * x;
}, 5)
```

*10*