FREE eBook

LEARNING

underscore.js

#underscore.js

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: underscore-js

It is an unofficial and free underscore.js ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official underscore.js.

Chapter 1: Getting started with underscore.js

Remarks

This section provides an overview of what underscore.js is, and why a developer might want to use it.

It should also mention any large subjects within underscore.js, and link out to the related topics. Since the Documentation for underscore.js is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Underscore is an open source functional programming utility library for JavaScript. Underscore provides many useful functions for working with arrays or collections of JavaScript objects, including filtering, sorting and querying.

Node.js

Make sure you have node and npm installed then type the following command

```
npm install underscore
```

Bower

Make sure you have node, npm and bower installed then type the following command

```
bower install underscore
```

NuGet

```
Install-Package underscore.js
```

Browser

Download the Development version or the Production version and link the script inside you HTML file using the `script` element.

```
<script src="js/underscore/underscore.js"></script>
```

Read Getting started with underscore.js online: https://riptutorial.com/underscore-js/topic/7688/getting-started-with-underscore-js

Chapter 2: Collections

Examples

map

The `.map` function accepts an array and an iteratee function, the iteratee produces a transformed copy of each array object.

The iteratee function provides 3 arguments

1. `item` - The current iterated object
2. `i` - The index of the iterated object
3. `list` - A reference to the original array/list

The new Array will be of the same length as the old array but will hold the trasformed objects

Example:

```
_.map([1, 2, 3, 4], function(item, i, list) {
    return (item*item);
});
// [1, 4, 9, 16]
```

A more concise way to write the above example using ES6 would be

```
_.map([1, 2, 3, 4], (item, i, list) => {
    return (item*item);
});
```

or using an inline lambda expression

```
_.map([1, 2, 3, 4], (item, i, list) => (item*item));
```

Map is also useful when you want to pluck properties from objects and make an array of them

Example:

```
let people = [{name: 'he-man', age: 22}, {name: 'man-at-arms', age: 44}];

_.map(people, function(item) {
    return item.name;
});
// ['he-man', 'man-at-arms']
```

each

The `_.each` function accepts an array or an object, an iteratee function and an optional `context` object, the iteratee function is invoked once and in order for each array item The iteratee function provides 3 arguments

```
item - The current iterated object (or value if an object was passed)
i - The index of the iterated object (or key if an object was passed)
list - A reference to the original array/list (the object that was passed)
```

Example 1:

```
_.each(["hello", "underscore"], function(item, i, list) {
    alert(item)
});
```

The above example will show 2 alerts, the first with the words "hello" and the second for "world".

Example 2:

```
_.each({one: 1, two: 2, three: 3}, (value, key, object) =>
    console.log(JSON.stringify(object));
);
```

This example will log a stringified version of the object 3 times.

`.each` is a terminal operation, and unlike other intermediate functions (map, pluck, values etc..) you don't need to return inside the iteratee function body.

Read Collections online: https://riptutorial.com/underscore-js/topic/7695/collections

Credits

S. No	Chapters	Contributors
1	Getting started with underscore.js	Ashiquzzaman, Community, svarog
2	Collections	svarog