

Take a nested Javascript object and flatten it, or unflatten an object with delimited keys.

## Installation

```
1 $ npm install flat
```

## Methods

### **flatten(original, options)**

Flattens the object - it'll return an object one level deep, regardless of how nested the original object was:

```
1 import { flatten } from 'flat'
2
3 flatten({
4   key1: {
5     keyA: 'valueI'
6   },
7   key2: {
8     keyB: 'valueII'
9   },
10  key3: { a: { b: { c: 2 } } }
11 })
12
13 // {
14 //   'key1.keyA': 'valueI',
15 //   'key2.keyB': 'valueII',
16 //   'key3.a.b.c': 2
17 // }
```

### **unflatten(original, options)**

Flattening is reversible too, you can call `unflatten` on an object:

```
1 import { unflatten } from 'flat'
2
3 unflatten({
4   'three.levels.deep': 42,
5   'three.levels': {
6     nested: true
7   }
8 })
```

---

```
7     }
8   })
9
10  // {
11  //   three: {
12  //     levels: {
13  //       deep: 42,
14  //       nested: true
15  //     }
16  //   }
17  // }
```

## Options

### delimiter

Use a custom delimiter for (un)flattening your objects, instead of `.`.

### safe

When enabled, both `flat` and `unflatten` will preserve arrays and their contents. This is disabled by default.

```
1  import { flatten } from 'flat'
2
3  flatten({
4    this: [
5      { contains: 'arrays' },
6      { preserving: {
7        them: 'for you'
8      } }
9    ]
10 }, {
11   safe: true
12 })
13
14 // {
15 //   'this': [
16 //     { contains: 'arrays' },
17 //     { preserving: {
18 //       them: 'for you'
19 //     } }
20 //   ]
21 // }
```

---

## object

When enabled, arrays will not be created automatically when calling unflatten, like so:

```
1 unflatten({
2   'hello.you.0': 'ipsum',
3   'hello.you.1': 'lorem',
4   'hello.other.world': 'foo'
5 }, { object: true })
6
7 // hello: {
8 //   you: {
9 //     0: 'ipsum',
10 //    1: 'lorem',
11 //   },
12 //   other: { world: 'foo' }
13 // }
```

## overwrite

When enabled, existing keys in the unflattened object may be overwritten if they cannot hold a newly encountered nested value:

```
1 unflatten({
2   'TRAVIS': 'true',
3   'TRAVIS.DIR': '/home/travis/build/kvz/environmental'
4 }, { overwrite: true })
5
6 // TRAVIS: {
7 //   DIR: '/home/travis/build/kvz/environmental'
8 // }
```

Without `overwrite` set to `true`, the `TRAVIS` key would already have been set to a string, thus could not accept the nested `DIR` element.

This only makes sense on ordered arrays, and since we're overwriting data, should be used with care.

## maxDepth

Maximum number of nested objects to flatten.

```
1 import { flatten } from 'flat'
2
3 flatten({
4   key1: {
```

---

```
5     keyA: 'valueI'
6   },
7   key2: {
8     keyB: 'valueII'
9   },
10  key3: { a: { b: { c: 2 } } }
11 }, { maxDepth: 2 })
12
13 // {
14 //   'key1.keyA': 'valueI',
15 //   'key2.keyB': 'valueII',
16 //   'key3.a': { b: { c: 2 } }
17 // }
```

### transformKey

Transform each part of a flat key before and after flattening.

```
1 import { flatten, unflatten } from 'flat'
2
3 flatten({
4   key1: {
5     keyA: 'valueI'
6   },
7   key2: {
8     keyB: 'valueII'
9   },
10  key3: { a: { b: { c: 2 } } }
11 }, {
12   transformKey: function(key){
13     return '__' + key + '__';
14   }
15 })
16
17 // {
18 //   '__key1__.__keyA__': 'valueI',
19 //   '__key2__.__keyB__': 'valueII',
20 //   '__key3__.__a__.__b__.__c__': 2
21 // }
22
23 unflatten({
24   '__key1__.__keyA__': 'valueI',
25   '__key2__.__keyB__': 'valueII',
26   '__key3__.__a__.__b__.__c__': 2
27 }, {
28   transformKey: function(key){
29     return key.substring(2, key.length - 2)
30   }
31 })
```

---

```
32
33 // {
34 //   key1: {
35 //     keyA: 'valueI'
36 //   },
37 //   key2: {
38 //     keyB: 'valueII'
39 //   },
40 //   key3: { a: { b: { c: 2 } } }
41 // }
```

## Command Line Usage

`flat` is also available as a command line tool. You can run it with `npx`:

```
1 npx flat foo.json
```

Or install the `flat` command globally:

```
1 npm i -g flat && flat foo.json
```

Accepts a filename as an argument:

```
1 flat foo.json
```

Also accepts JSON on stdin:

```
1 cat foo.json | flat
```