

Get started

Open in app

To make Medium work, we log user data.
By using Medium, you agree to our
Privacy Policy, including cookie policy.



BetterProgramming

Follow

204K Followers



How to Merge Deeply Nested Objects in JavaScript

The way to correctly merge nested objects



Yogesh Chavan May 19, 2020 · 2 min read



Photo by [Émile Perron](#) on [Unsplash](#).

There are many scenarios where we need to merge two objects that may be deeply nested.

In this article, we will see how to handle that correctly.

[Get started](#)[Open in app](#)

To make Medium work, we log user data.
By using Medium, you agree to our
Privacy Policy, including cookie policy.



```
1  const user = {
2    name: "David",
3    phone: 122345678,
4    location: {
5      city: "Camden",
6      country: "UK"
7    }
8  };
```

userobject.js hosted with ❤ by GitHub

[view raw](#)

And we want to update this object with the new values. For example, when updating their user profile, the user enters new values, so we need to merge the previous object with the new updated object.

If the user updates only some values, we will have an object like this:

```
1  const updates = {
2    name: "David",
3    phone: 123456789,
4    location: {
5      city: "Smithfield"
6    }
7  };
```

userupdate.js hosted with ❤ by GitHub

[view raw](#)

As you can see, the `user` object has a `location` property with the `city` and `country` properties inside it. But in the `updates` object, we don't have the `country` property, so how can we merge these objects?

Let's try using some known ways.

Using Object.assign

```
1  const user = {
2    name: 'David',
3    phone: 122345678,
```

[Get started](#)[Open in app](#)

To make Medium work, we log user data.
By using Medium, you agree to our
Privacy Policy, including cookie policy.



```
7   }
8   };
9   const updates = {
10    name: 'David',
11    phone: 123456789,
12    location: {
13      city: 'Smithfield'
14    }
15  };
16  console.log(Object.assign({}, user, updates));
17  /* output:
18  {
19    name: 'David',
20    phone: 123456789,
21    location: {
22      city: 'Smithfield'
23    }
24  }
25  */
```

objectassign.js hosted with ❤ by GitHub

[view raw](#)

As you can see, the `location` property from the `update` object has completely replaced the previous `location` object, so the `country` property is lost.

This is because `Object.assign` does a shallow merge and not a deep merge. A shallow merge means it will merge properties only at the first level and not the nested level.

Using Spread Operator

```
1  const user = {
2    name: 'David',
3    phone: 122345678,
4    location: {
5      city: 'Camden',
6      country: 'UK'
7    }
8  };
9  const updates = {
```

[Get started](#)[Open in ap](#)

To make Medium work, we log user data.
By using Medium, you agree to our
Privacy Policy, including cookie policy.



```
13   city: 'Smithfield'
14   }
15 };
16 console.log({ ...user, ...updates });
17 /* output:
18 {
19   name: 'David',
20   phone: 123456789,
21   location: {
22     city: 'Smithfield'
23   }
24 }
25 */
```

spreadoperator.js hosted with ❤ by GitHub

[view raw](#)

As you can see, this also produces the same result because `Object.assign` and the spread operator just shallow-merge the objects.

To fix this and correctly merge two deeply nested objects, we can use the `merge` method provided by the Lodash library.

Using Lodash merge

```
1  const user = {
2    name: "David",
3    phone: 122345678,
4    location: {
5      city: "Camden",
6      country: "UK"
7    }
8  };
9  const updates = {
10   name: "David",
11   phone: 123456789,
12   location: {
13     city: "Smithfield"
14   }
15 };
16 console.log(_.merge(user, updates));
```

[Get started](#)[Open in app](#)

To make Medium work, we log user data.
By using Medium, you agree to our
Privacy Policy, including cookie policy.



```
20    phone: 123456789,  
21    location: {  
22      country: 'UK',  
23      city: 'Smithfield'  
24    }  
25  }  
26  */
```

lodashmerge.js hosted with ❤ by GitHub

[view raw](#)

Now, it works as expected and the `country` property is not lost in the merge.

Lodash is the best utility library that has lots of functions for real-life applications. To explore other useful `lodash` methods, check out [my previous article](#).

That's it for today. I hope you learned something new.

Don't forget to subscribe to get my weekly newsletter with amazing tips, tricks, and articles directly in your inbox [here](#).

Sign up for programming bytes

By Better Programming

A monthly newsletter covering the best programming articles published across Medium. Code tutorials, advice, career opportunities, and more! [Take a look](#).

[Get this newsletter](#)

[Programming](#) [JavaScript](#) [React](#) [Reactjs](#) [Nodejs](#)

[About](#) [Write](#) [Help](#) [Legal](#)

Get started

Open in ap

To make Medium work, we log user data.
By using Medium, you agree to our
Privacy Policy, including cookie policy.

