

How to Merge Deeply Nested Objects in JavaScript

The way to correctly merge nested objects



Yogesh Chavan May 19, 2020 · 2 min read

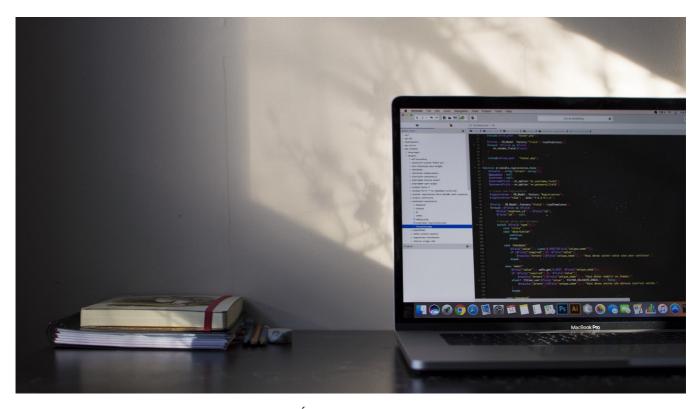


Photo by <u>Émile Perron</u> on <u>Unsplash</u>.

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.

```
To make Medium work, we log user data.
 Get started
                  Open in ap
                                By using Medium, you agree to our
                                Privacy Policy, including cookie policy.
    const user = {
1
      name: "David",
2
3
      phone: 122345678,
      location: {
4
         city: "Camden",
         country: "UK"
7
      }
    };
8
userobject.js hosted with \heartsuit 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 ap

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

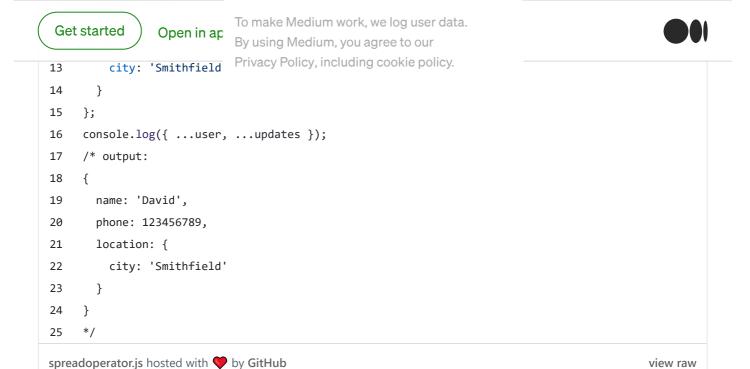
```
7
       }
     };
    const updates = {
9
      name: 'David',
10
      phone: 123456789,
11
       location: {
12
         city: 'Smithfield'
13
14
       }
15
     };
16
     console.log(Object.assign({}, user, updates));
     /* output:
17
18
      name: 'David',
19
       phone: 123456789,
20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

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

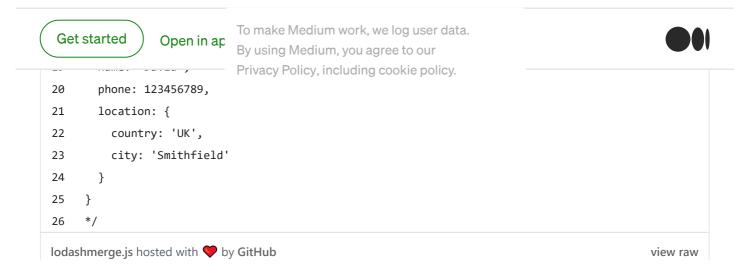


As you can see, this also produces the same result because <code>object.assign</code> and the <code>spread operator</code> just shallow-merge the objects.

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

Using Lodash merge

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



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 <code>lodash</code> methods, check out <u>my previous article</u>.

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 <u>here.</u>

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! <u>Take a look.</u>

Get this newsletter

Programming JavaScript React Reactjs Nodejs



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

