REACT

What are three dots (...) or spread operators in React

Jan 8, 2022 Abhishek EH 4 Min Read

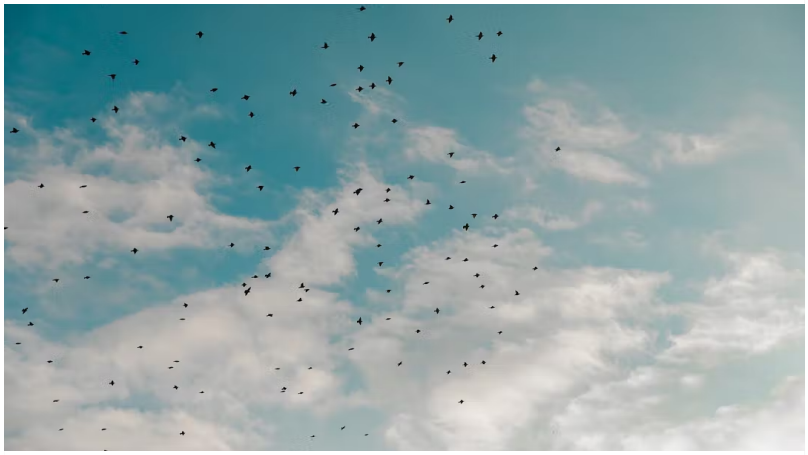


Table of Contents

You might have often seen code in React where they have used three dots (`...`) ahead of a variable, as shown below:

jsx

```jsx
<Component {...props} />
```

These are called spread operators in JavaScript and this article, we will see the different use cases of how spread operators can be used in React and JavaScript.

Passing props using the spread operator

Say you have a component called `Person` and you want to pass three different properties, `firstName` , `lastName` , and `age` . Traditionally we would pass them as

```
import React from "react"

export const Person = () => {
  const { firstName, lastName, age } = p
  return (
    <div>
      <p>
        Name :{firstName} {lastName}
      </p>
      <p>Age :{age}</p>
    </div>
  )
}

const App = () => {
  const person = { firstName: "John", la
  return (
    <Person
      firstName={person.firstName}
      lastName={person.lastName}
      age={person.age}
    />
  )
}

export default App
```

Notice that we will access each property and write it individually. As the number of properties grow, the code becomes bulky. We can use spread operator here to pass all the properties inside the person object as shown below:

```
export const Person = () => {
  const { firstName, lastName, age } = p
  return (
    <div>
      <p>
        Name :{firstName} {lastName}
      </p>
      <p>Age :{age}</p>
    </div>
  )
}

const App = () => {
  const person = { firstName: "John", las
  return <Person {...person} />
}

export default App
```

Arrays and spread operator

Spread operator can also be used for performing different array operations

Creating a copy of an array

We can copy items for an array to another array using spread operator as shown below:

js

```
const arr1 = [1, 2, 3]
```

```
5  console.log(arr2) // [1, 2, 4]
```

Note that the original array will not be impacted when we modify the copy of the array. Also, note that it does a shallow copy, that is, it copies only the items at the top level by value and all the nested properties will be copied by reference.

Combining 2 arrays

We can combine 2 arrays and create a new array as shown below, using spread operators:

js

```
1  const arr1 = [1, 2, 3]
2  const arr2 = [4, 5, 6]
3  const arr3 = [...arr1, ...arr2]
4  console.log(arr3) // [1, 2, 3, 4, 5, 6]
```

Adding items to an array

While copying an array, we can add an element to the beginning or to the end:

js

```
1  const arr1 = [1, 2, 3]
2  const arr2 = [0, ...arr1]
3  const arr3 = [...arr1, 4]
4  console.log(arr2) // [0, 1, 2, 3]
5  console.log(arr3) // [1, 2, 3, 4]
```

In my previous article, I have explained how to add an item in between the array.

Passing arrays to a function

We can use spread operator to pass an array to a function as separate arguments:

js

```
const sum = (a, b, c) => {
  return a + b + c
}

const arr1 = [1, 2, 3]

sum(...arr1)
```

Spread operator for Object operations

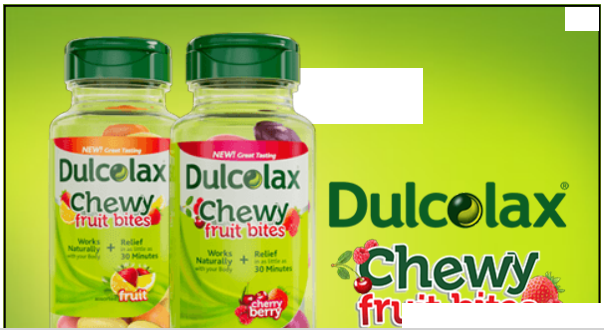
Now, let's see different object operations that can be performed using the spread operator.

Copying items of an object

Similar to arrays, we can create shallow copy of an object:

js

```
const obj1 = { firstName: "John", lastNa
const person = { ...obj1 }
console.log(person) // {firstName: 'John
```



Combining 2 objects

Similar to arrays, we can combine 2 objects as shown below:

js

```
const obj1 = { firstName: "John" }
const obj2 = { lastName: "Doe", age: 32 }
const person = { ...obj1, ...obj2 }
console.log(person) // {firstName: 'John
```

> If the same property is present in both the objects, then the property of the left object will be replaced by the property of the right object in the created copy.

Adding a prop while copying the object

We can add additional properties while copying the object:

js

```
const obj1 = { firstName: "John", lastNa
const person = { ...obj1, age: 32, profe
console.log(person) // {firstName: 'John
```

Updating existing properties of the object

Similar to adding props, we can update existing properties as well:

js

```
const obj1 = {
  firstName: "John",
  lastName: "Doe",
  age: 32,
  profession: "Full Stack Developer",
}
const person = { ...obj1, age: 33 }
console.log(person) //{firstName: 'John'
```

Do follow me on twitter where I post developer insights more often!

Leave a Comment

Message

Your comment

Name

Your Name

E-mail

Your Email (won't be p

Post Comment