Time to switch gears for a moment and check out the spread operator!

**Spread operator**

The **spread operator**, written with three consecutive dots ( ... ), is new in ES6 and gives you the ability to expand, or *spread*, [iterable objects](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Iterators_and_Generators" \l "Iterators" \t "_blank) into multiple elements.

Let’s take a look at a few examples to see how it works.

**const** books = ["Don Quixote", "The Hobbit", "Alice in Wonderland", "Tale of Two Cities"];

console.log(...books);

***Prints:****Don Quixote The Hobbit Alice in Wonderland Tale of Two Cities*

**const** primes = **new** Set([2, 3, 5, 7, 11, 13, 17, 19, 23, 29]);

console.log(...primes);

***Prints:****2 3 5 7 11 13 17 19 23 29*

If you look at the output from the examples, notice that both the array and set have been expanded into their individual elements. So how is this useful?

***NOTE:****Sets are a new built-in object in ES6 that we’ll cover in more detail in a future lesson.*

**Combining arrays with concat**

One example of when the spread operator can be useful is when combining arrays.

If you’ve ever needed to combine multiple arrays, prior to the spread operator, you were forced to use the Array’s concat() method.

**const** fruits = ["apples", "bananas", "pears"];

**const** vegetables = ["corn", "potatoes", "carrots"];

**const** produce = fruits.concat(vegetables);

console.log(produce);

***Prints:****["apples", "bananas", "pears", "corn", "potatoes", "carrots"]*

This isn’t terrible, but wouldn’t it be nice if there was a shorthand way to write this code?

For example, something like…

***⚠️ Upcoming const Warning ⚠️***

*If you're following along by copy/pasting the code, then you've already declared the produce variable with the const keyword. The following code will try to redeclare and reassign the variable, so depending on how you're running the code, it might throw an error.*

*Remember that variables declared with const cannot be redeclared or reassigned in the same scope.*

**const** produce = [fruits, vegetables];

console.log(produce);

***Prints:****[Array[3], Array[3]]*

Unfortunately, that doesn’t work.

Instead of combining both arrays, this code actually adds the fruits array at the first index and the vegetables array at the second index of the produce array.

How about trying the spread operator?

/\*

\* Instructions: Use the spread operator to combine the `fruits` and `vegetables` arrays into the `produce` array.

\*/

const fruits = ["apples", "bananas", "pears"];

const vegetables = ["corn", "potatoes", "carrots"];

const produce = [...fruits, ...vegetables];

console.log(produce);