# How Do You Update Arrays in State?

freecodecamp.org/learn/front-end-development-libraries-v9/lecture-working-with-state-and-responding-to-events-in-react/how-do-you-update-arrays-in-state


In React, updating arrays in state is quite straightforward, but it can be easy to make a mistake, especially if you're coming from vanilla JavaScript where you can modify arrays directly.

In React, state is treated as immutable so it can recognize changes and make the proper updates to the user interface.

Let's look at how you can update arrays held in state in React.

One of the most common mistakes when updating arrays in a React state is to directly modify the array using methods like `push()`, `pop()`, or `splice()`. These methods mutate the original array, and React does not allow that.

React relies on a new array reference to detect changes, so directly modifying the array can prevent the component from re-rendering as expected.

Here's an example of using the `push()` method to add to an array in state, which won't work:

```
import { useState } from "react";

function ItemsList() {
  const [items, setItems] = useState([
    { id: 0, name: "Item 1" },
    { id: 1, name: "Item 2" },
    { id: 2, name: "Item 3" },
  ]);

  const addItem = () => {
    const newItem = { id: items.length + 1, name: `Item ${items.length + 1}` };
    items.push(newItem); // This modifies the original array
    setItems(items); // React will not detect this change
  };

  return (
    <div>
      <button onClick={addItem}>Add Item</button>
      <ul>
        {items.map((item) => (
          <li key={item.id}>{item.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default ItemsList;
```

If you click the `Add Item` button, nothing happens in the user interface.

It might also be tempting to remove items from the array with the `pop()` method:

```
const removeItem = () => {
  items.pop(); // Modifies the original array
  setItems(items); // React will not detect this change, either
};
```

To update an array in state, the key is to create a new array, do your operations, and pass that to React, rather than mutate the existing array.

Because it's a new array, React will know that the state has been changed, and trigger a re-render.

Here's how you can add to the `items` array using the spread operator:

```
const addItem = () => {
  const newItem = {
    id: items.length + 1,
    name: `Item ${items.length + 1}`,
  };

  // Creates a new array
  setItems((prevItems) => [...prevItems, newItem]);
};
```

`[...prevItems, newItem]` creates a new array by copying all items in the existing `items` array held in state, then adds `newItem` at the end, which increments the `id` and the item number.

If you want to remove something from the array, you can use the `filter()` method, which returns a new array after filtering out whatever you want to remove:

```
const removeItem = (id) => {
  setItems((prevItems) => prevItems.filter((item) => item.id !== id));
};
```

Here's the full code:

```
import { useState } from "react";

function ItemsList() {
  const [items, setItems] = useState([
    { id: 0, name: "Item 1" },
    { id: 1, name: "Item 2" },
    { id: 2, name: "Item 3" },
  ]);

  const addItem = () => {
    const newItem = { id: items.length + 1, name: `Item ${items.length + 1}` };
    setItems((prevItems) => [...prevItems, newItem]); // Creates a new array
  };

  const removeItem = (id) => {
    setItems((prevItems) => prevItems.filter((item) => item.id !== id)); // Creates a
new array
  };

  return (
    <div>
      <button onClick={addItem}>Add Item</button>
      <ul>
        {items.map((item) => (
          <li key={item.id}>
            {item.name}{" "}
            <button onClick={() => removeItem(item.id)}>Remove</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default ItemsList;
```

Those are some common ways you can update an array in state.

## Questions

# Which of the following is the correct way to update state and add new items to an array?

```
setItems(copy [prevItems,
newItem]);
```

```
setItems(new Array = [...prevItems,
newItem]);
```

```
items = (prevItems) => [...prevItems,
newItem];
```

```
setItems((prevItems) => [...prevItems,
newItem]);
```

# Why should you NOT directly modify an array in React state?

It automatically clears the array.

It adds unwanted duplicates to the array.

It causes errors in the component.

It prevents React from detecting changes.

# Which method is best for removing an item from an array in React state?

`splice()`

`push()`

`pop()`

`filter()`