🔀 1. Merging Arrays

✅ Using Spread (...)

javascript Copy Edit

```javascript
const a = [1, 2]; const b = [3, 4]; const merged = [...a, ...b]; console.log(merged); // [1, 2, 3, 4]
```

✅ With `.concat()` (older way)

javascript Copy Edit

```javascript
const merged = a.concat(b);
```

🔀 2. Merging Objects

✅ Using Spread (...)

javascript Copy Edit

```javascript
const user = { name: "Alice", age: 25 }; const update = { age: 26, city: "New York" }; const mergedUser = { ...user, ...update }; console.log(mergedUser); // { name: "Alice", age: 26, city: "New York" }
```

> 👉 **If both objects have the same key, the last one wins (`update.age` overwrites `user.age`).**

✅ Using `Object.assign()`

javascript Copy Edit

```javascript
const mergedUser = Object.assign({}, user, update);
```

Both `Object.assign()` and spread do the same thing — the spread operator is just cleaner.

🧠 In React

Merging is super useful when updating **state** that's an array or object.

➕ Merge Objects in State

jsx

Copy Edit

```jsx
setUser(prev => ({ ...prev, city: "New York" }));
```

➕ Merge Arrays in State

jsx

Copy Edit

```jsx
setItems(prev => [ ...prev, "new item" ]);
```

🔧 When to Use Merging

Task	Method	Example
Combine two arrays	`[...]` or `.concat()`	`[...a, ...b]`
Merge object updates	`...` or `Object.assign()`	`{...a, ...b}`
Update part of state (React)	`...`	`setState(prev => ({...prev, key: val}))`