React: The Virtual DOM

Fighting Wasteful DOM Manipulation

The Problem

DOM manipulation is the heart of the modern, interactive web. Unfortunately, it is also a lot slower than most JavaScript operations.

This slowness is made worse by the fact that **most JavaScript frameworks update the DOM much more than they have to.**

As an example, let's say that you have a list that contains ten items. You check off the first item. Most JavaScript frameworks would rebuild *the entire list*. That's ten times more work than necessary! Only one item changed, but the remaining nine get rebuilt exactly how they were before.

Rebuilding a list is no big deal to a web browser, but modern websites can use huge amounts of DOM manipulation. Inefficient updating has become a serious problem.

To address this problem, the people at React popularized something called the *virtual DOM.*

The Virtual DOM

In React, for every DOM object, there is a corresponding "virtual DOM object." A virtual DOM object is a *representation* of a DOM object, like a lightweight copy.

A virtual DOM object has the same properties as a real DOM object, but it lacks the real thing's power to directly change what's on the screen.

Manipulating the DOM is slow. Manipulating the virtual DOM is much faster, because nothing gets drawn onscreen. Think of manipulating the virtual DOM as editing a blueprint, as opposed to moving rooms in an actual house.

How it helps

When you render a JSX element, every single virtual DOM object gets updated.

This sounds incredibly inefficient, but the cost is insignificant because the virtual DOM can update so quickly.

Once the virtual DOM has updated, then React compares the virtual DOM with a virtual DOM *snapshot* that was taken right before the update.

By comparing the new virtual DOM with a pre-update version, React figures out *exactly which virtual DOM objects have changed.* This process is called "diffing."

Once React knows which virtual DOM objects have changed, then React updates those objects, *and only those objects,* on the real DOM. In our example from earlier, React would be smart enough to rebuild your one checked-off list-item, and leave the rest of your list alone.

This makes a big difference! React can update only the necessary parts of the DOM. React's reputation for performance comes largely from this innovation.

In summary, here's what happens when you try to update the DOM in React:

1. The entire virtual DOM gets updated.
2. The virtual DOM gets compared to what it looked like before you updated it. React figures out which objects have changed.
3. The changed objects, and the changed objects only, get updated on the *real* DOM.
4. Changes on the real DOM cause the screen to change.

If you'd like to learn more about the virtual DOM, here's a good place to start.