Desmond Nyamador

React's Virtual DOM Explained

Desmond Nyamador

Oct 23, 2020 • 4 Min read • 345 Views

Oct 23, 2020 • 4 Min read • 345 Views

Web Development Front End Web Dev... Client-side Framew... React

Introduction

9

Introduction

You may have built a couple of React apps, or maybe you're just getting started. But a big question no matter your level of experience is, how does React decide which component to re-render and how does it do so efficiently? In this guide, you'll learn about React's virtual DOM how it works, including other concepts such as reconciliation and fibers.

Virtual DOM vs. Regular DOM

First off, the virtual DOM (VDOM) is not the same as the shadow DOM. Rather, the virtual DOM stores a representation of the UI in memory and is synced with the actual DOM with the help of React DOM. The process of syncing the real *DOM* with the *VDOM* is referred to as *reconciliation.* Internally, React uses objects called *fibers* to keep more details of the component tree. Do you recognize the snippet below? The `render()` function creates a tree of components that is used in the reconciliation process.

jsx

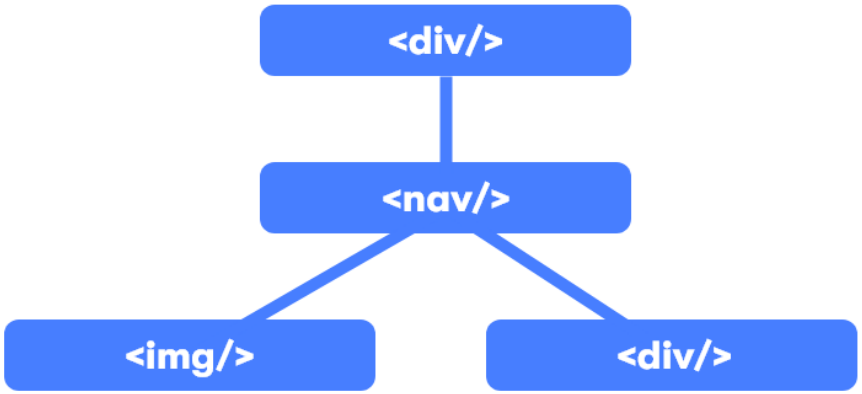
```jsx
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(
    <App/>
    document.getElementById('root')
);
```

Using the VDOM, it becomes unnecessary to manipulate attributes of DOM elements manually and update the actual DOM. A typical example of why React's method of solving this is so great is rendering a list of items. In reality, only one item might have changed in the list. If the list contains 100 items, 99 nodes that never changed would re-render. That is very inefficient!

Reconciliation in React

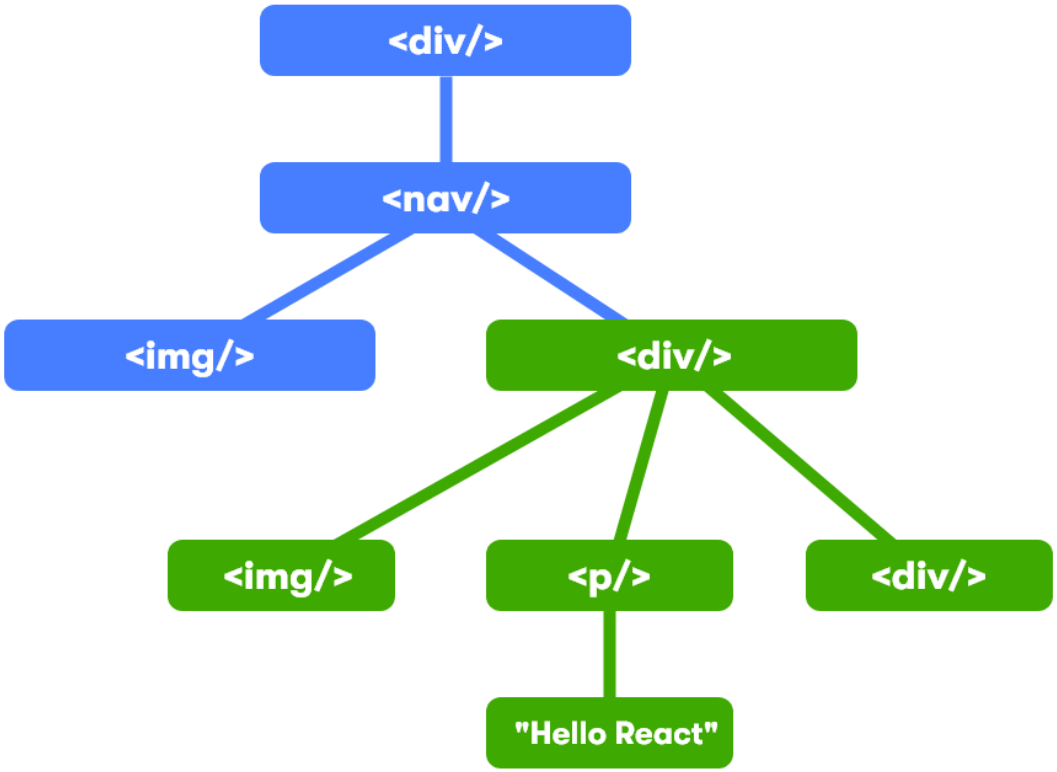
As indicated above, reconciliation is the process of syncing the VDOM with the real DOM. For this to happen, React creates a tree starting from the root node. The diagram below shows the tree representation of the real DOM consisting of four nodes.



Real DOM

Considering that the app's state changes, React uses its diffing algorithm (very similar to how Git compares changes in files) to compare the root elements in the virtual DOM and real DOM.

Whenever it encounters root elements that have changed, it tears down the nodes whose states have changed and remounts them. In this example, the nodes marked green in the tree representation below are the ones that get remounted.



Virtual DOM

Keys and Attributes

To help React figure out which items in a list to mount, it requires you to add a key attribute to every list item. With the code snippet below, React would be able to figure out which element in the list has changed and mount it. The unique identifier provided by you helps React identify the elements. This can be from an id retrieved from an API or otherwise.

jsx

```jsx
<ul>
  <li key="1">Skills</li>
  <li key="2">Projects</li>
</ul>

<ul>
  <li key="0">Flow</li>
  <li key="1">Skills</li>
  <li key="2">Projects</li>
</ul>
```

Attributes

In the code block below, React only changes the attributes of the DOM node and not the underlying node itself. This is why it's very efficient!

jsx

```
<div className="blue">
<div className="green">
```

Conclusion

And that's it! You're a React guru now. If you'd like to read more on reconciliation or the Virtual DOM in React, visit the following sections of the React docs:

1. https://reactjs.org/docs/faq-internals.html
2. https://reactjs.org/docs/reconciliation.html

You can also check out Saravanan Dhandapani's guide on how React's virtual DOM sets it apart from frontend frameworks and libraries:

Virtual DOM - the Difference Maker in React JS

Feel free to ping me on Twitter if you'd like to chat more at @DesmondNyamador.