**Real DOM**

First things first, DOM stands for “Document Object Model”. The DOM in simple words represents the UI of your application. Everytime there is a change in the state of your application UI, the DOM gets updated to represent that change. Now the catch is frequently manipulating the DOM affects performance, making it slow.

**What makes DOM manipulation slow?**

The DOM is represented as a tree data structure. Because of that, the changes and updates to the DOM are fast. But after the change, the updated element and it’s children have to be re-rendered to update the application UI. The re-rendering or re-painting of the UI is what makes it slow. Therefore, the more UI components you have, the more expensive the DOM updates could be, since they would need to be re-rendered for every DOM update.

**Virtual DOM**

That’s where the concept of virtual DOM comes in and performs significantly better than the real DOM. The virtual DOM is only a virtual representation of the DOM. Everytime the state of our application changes, the virtual DOM gets updated instead of the real DOM.

Well, you may ask ” Isn’t the virtual DOM doing the same thing as the real DOM, this sounds like double work? How can this be faster than just updating the real DOM?”

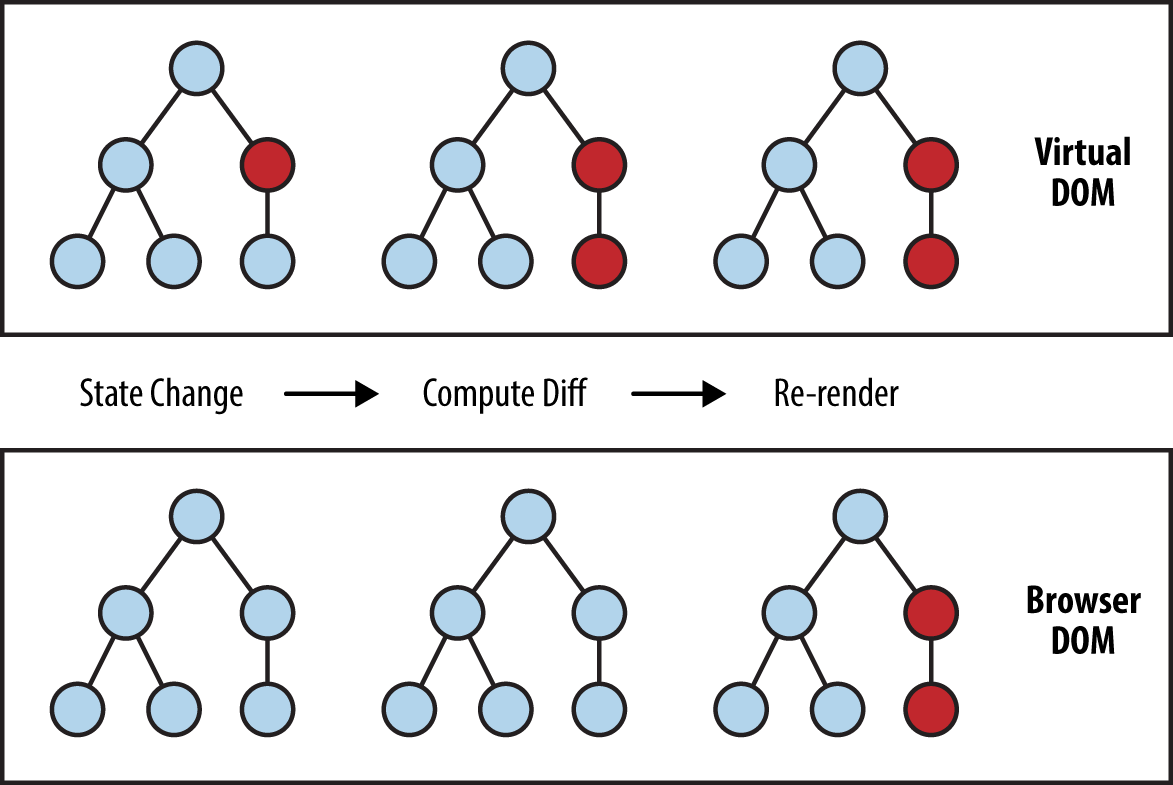
The answer is virtual DOM is much faster and efficient, here is why.

**How is Virtual DOM faster?**

When new elements are added to the UI, a virtual DOM, which is represented as a tree is created. Each element is a node on this tree. If the state of any of these elements changes, a new virtual DOM tree is created. This tree is then compared or “diffed” with the previous virtual DOM tree.

Once this is done, the virtual DOM calculates the best possible method to make these changes to the real DOM. This ensures that there are minimal operations on the real DOM. Hence, reducing the performance cost of updating the real DOM.

The image below shows the virtual DOM tree and the diffing process.



**source: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch02.html**

The red circles represent the nodes that have changed. These nodes represent the UI elements that have had their state changed. The difference between the previous version of the virtual DOM tree and the current virtual DOM tree is then calculated. The whole parent subtree then gets re-rendered to give the updated UI. This updated tree is then batch updated to the real DOM.

**How does React use Virtual DOM**

Now that you have a fair understanding of what a Virtual DOM is, and how it can help with performance of your app, lets look into how React leverages the virtual DOM.

In React every UI piece is a component, and each component has a state. React follows the observable pattern and listens for state changes. When the state of a component changes, React updates the virtual DOM tree. Once the virtual DOM has been updated, React then compares the current version of the virtual DOM with the previous version of the virtual DOM. This process is called “diffing”.

Once React knows which virtual DOM objects have changed, then React updates **only**those objects, in the real DOM. This makes the performance far better when compared to manipulating the real DOM directly. This makes React standout as a high performance JavaScript library.

In simple words, you tell React what state you want the UI to be in, and it makes sure that the DOM matches that state. The great benefit here is that as a developer, you would not need to know how the attribute manipulation, event handling or the manual DOM updates happen behind the scenes.

All of these details are abstracted away from React developers. All you need to do is update the states of your component as and when needed and React takes care of the rest. This ensures a superior developer experience when using React.

**React render() function**

*render()* is where the UI gets updated and rendered. *render()* is the required lifecycle method in React. You can learn more about React lifecycle methods in detail from my [blog post](https://programmingwithmosh.com/javascript/react-lifecycle-methods/).

*render()* function is the point of entry where the tree of React elements are created. When a *state* or *prop* within the component is updated, the *render()* will return a different tree of React elements. If you use *setState()*within the component, React immediately detects the state change and re-renders the component.

React then figures out how to efficiently update the UI to match the most recent tree changes.

This is when React updates its virtual DOM first and updates only the object that have changed in the real DOM.

**Batch Update**

React follows a batch update mechanism to update the real DOM. Hence, leading to increased performance. This means that updates to the real DOM are sent in batches, instead of sending updates for every single change in state.

The repainting of the UI is the most expensive part, and React efficiently ensures that the real DOM receives only batched updates to repaint the UI.

**Recap**

* Frequent DOM manipulations are expensive and performance heavy.
* Virtual DOM is a virtual representation of the real DOM.
* When state changes occur, the virtual DOM is updated and the previous and current version of virtual DOM is compared. This is called “diffing”.
* The virtual DOM then sends a batch update to the real DOM to update the UI.
* React uses virtual DOM to enhance its performance.
* It uses the observable to detect state and prop changes.
* React uses an efficient diff algorithm to compare the versions of virtual DOM.
* It then makes sure that batched updates are sent to the real DOM for repainting or re-rendering of the UI.