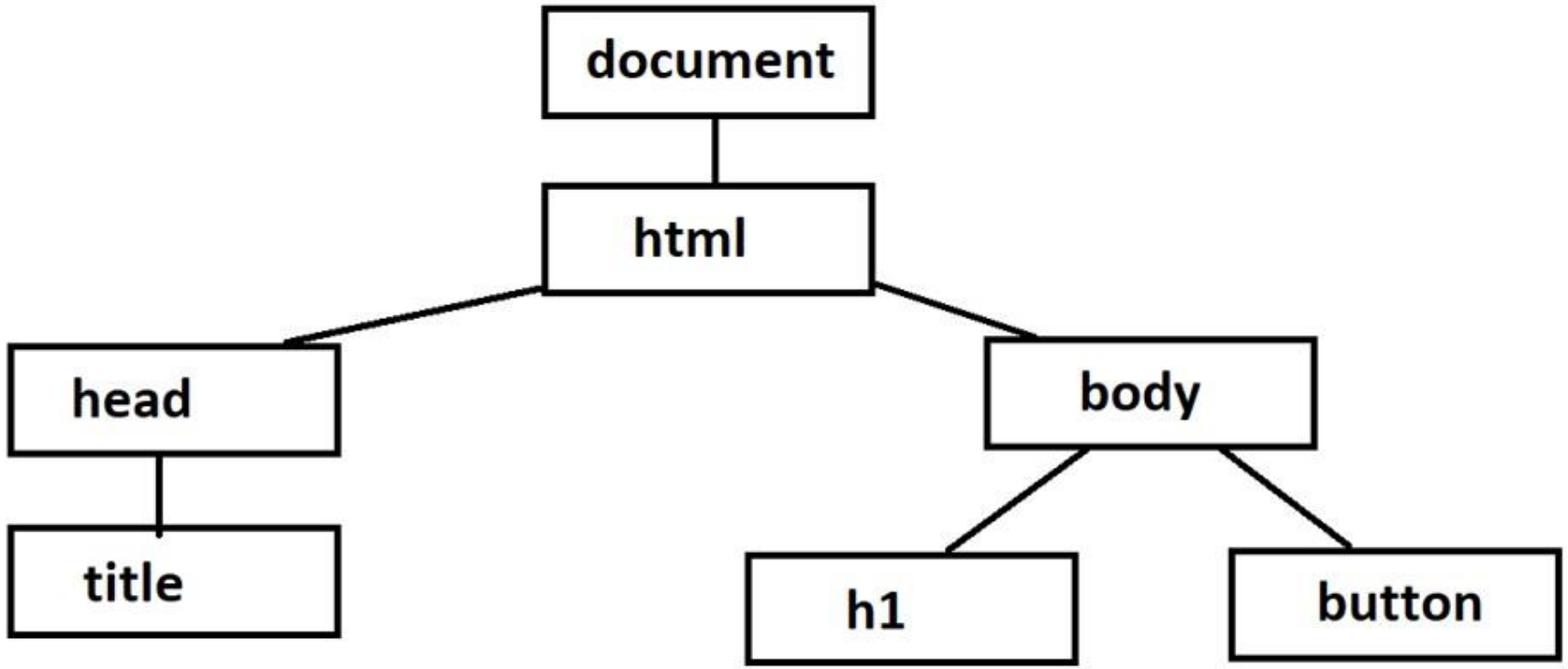
Why Does React use Virtual DOM? What is Reconciliation?

DOM stands for **Document Object Model** where **Document** is XML/HTML file, **Object** is the tags/elements of that file, **model** is the layout structure .DOM represents the content of XML or HTML document as a tree structure. When an HTML document is loaded into a browser, that **document** becomes an **object**, the DOM is created automatically by the browser and each HTML element is represented as an **object** in the DOM. The **document** object is the root node of the HTML document. DOM is an API which can be used with JavaScript or other languages to access and manipulate elements of HTML documents. In the DOM all HTML elements are objects, DOM has a tree-like structure which is a representation of HTML document as nodes/objects. There are built in DOM methods such as getAttribute, appendChild, addEventListener and properties such as innerText, innerHTML, lastChild. To access any element in an HTML page, we always start with the document object. To get an element by id, we can use getElementById(). DOM API is part of Web API, DOM API is the API of the browser, it is not part of JavaScript. The **document** object is created by the browser's rendering engine(Blink/Gecko) when it parses an HTML document. After the DOM is created, the document object provides an API to programming languages such as JavaScript to manipulate and access the contents of the web page. The **window** object is also created by the browser's rendering engine. The window object provides a way to interact with the browser window or tab.

DOM Tree representation




```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>shop</title>
</head>
<body>
    <h1>Welcome</h1>
    <button>Login</button>
</body>
</html>
```

For each element in the **html** file we see on the top right, an object will be created by the browser and we will have a DOM tree structure seen above on the left side. The document object always comes at the top. Each object in the tree is a node.

React uses virtual DOM which is a representation of the real DOM in memory. React compares the difference between the virtual and real DOM.

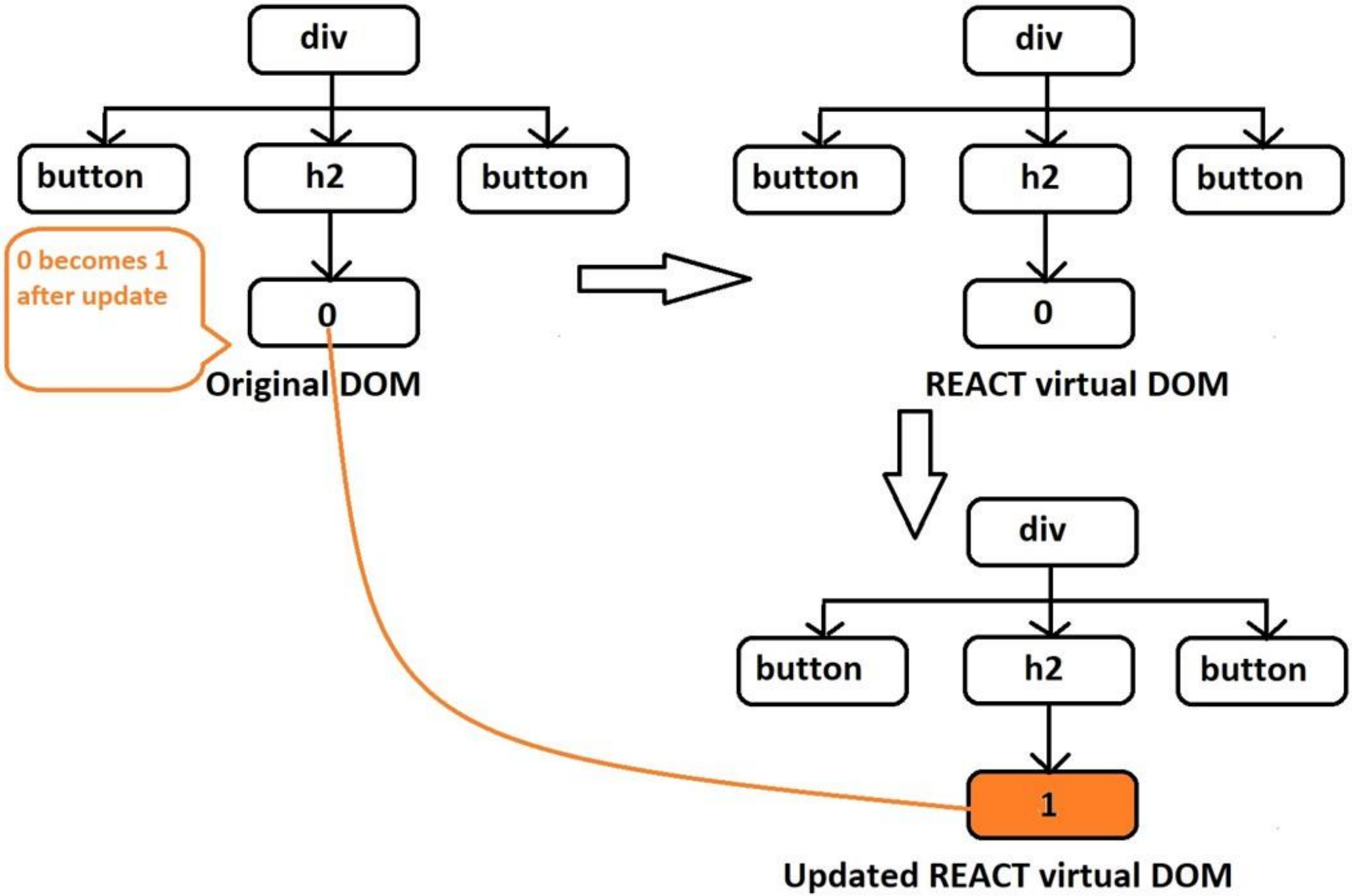
React does not update the original DOM tree directly. For every DOM tree representation, react creates one object that looks exactly like that original DOM tree, so that copy of the original DOM is called **virtual DOM**. So virtual DOM is the copy of original DOM in memory which react updates directly which then updates the original DOM partially. Why is this necessary? Browser's DOM has no mechanism to compare what has changed in the DOM tree so browsers re-render all the DOM nodes which is slower and memory inefficient. Virtual DOM is more efficient because the data stored/manipulated in the memory only, remember, in virtual DOM nothing gets drawn onscreen unlike browser's DOM.

If in the virtual DOM when an element's state gets changed, react does not update the virtual DOM directly but creates a new copy of the virtual DOM so second copy of the virtual DOM with the changes happened.

```
function Demo(){
        let[counter,setCounter] = useState(0);

        return <div>
                <button onClick={setCounter(--counter)}>Decrease</button>
                <h2>{counter}</h2>
                <button onClick={setCounter(++counter)}>Increase</button>
        </div>
}
```

When the react app is rendered for the first time, a virtual DOM tree is created in the memory behind the scenes. The same DOM tree is also rendered in the real DOM(In the browser), If a user clicks on the Increase button on the web page, the state will change and counter will be 1 and a new updated virtual DOM is created and will be re-rendered and we will have the updated value of counter 1. Updated React virtual DOM will check the difference between the previous virtual DOM and the current virtual DOM and calculates the difference, then the real DOM is updated. So in the case above only the **h2** is updated in the real DOM and in the real DOM, the complete DOM object will not be re-rendered but only the changed part is re-rendered. So as a result, once the virtual Dom is updated then React compares the current updated virtual DOM with the previous one, by comparing the new virtual DOM with the pre updated virtual DOM, React figures out exactly which virtual DOM objects have changed, using the **diffing algorithm,** after that, React updates the original DOM with those changes only. The process to diff one tree with another to identify which parts need to be changed and then update the original DOM with it is called **Reconciliation**.


div
button
h2
button
0 becomes 1 after update
0
Original DOM
div
button
h2
button
0
REACT virtual DOM
div
button
h2
button
1
Updated REACT virtual DOM