Skills Network

Virtual DOM Manipulation in React

IBM

What you will learn

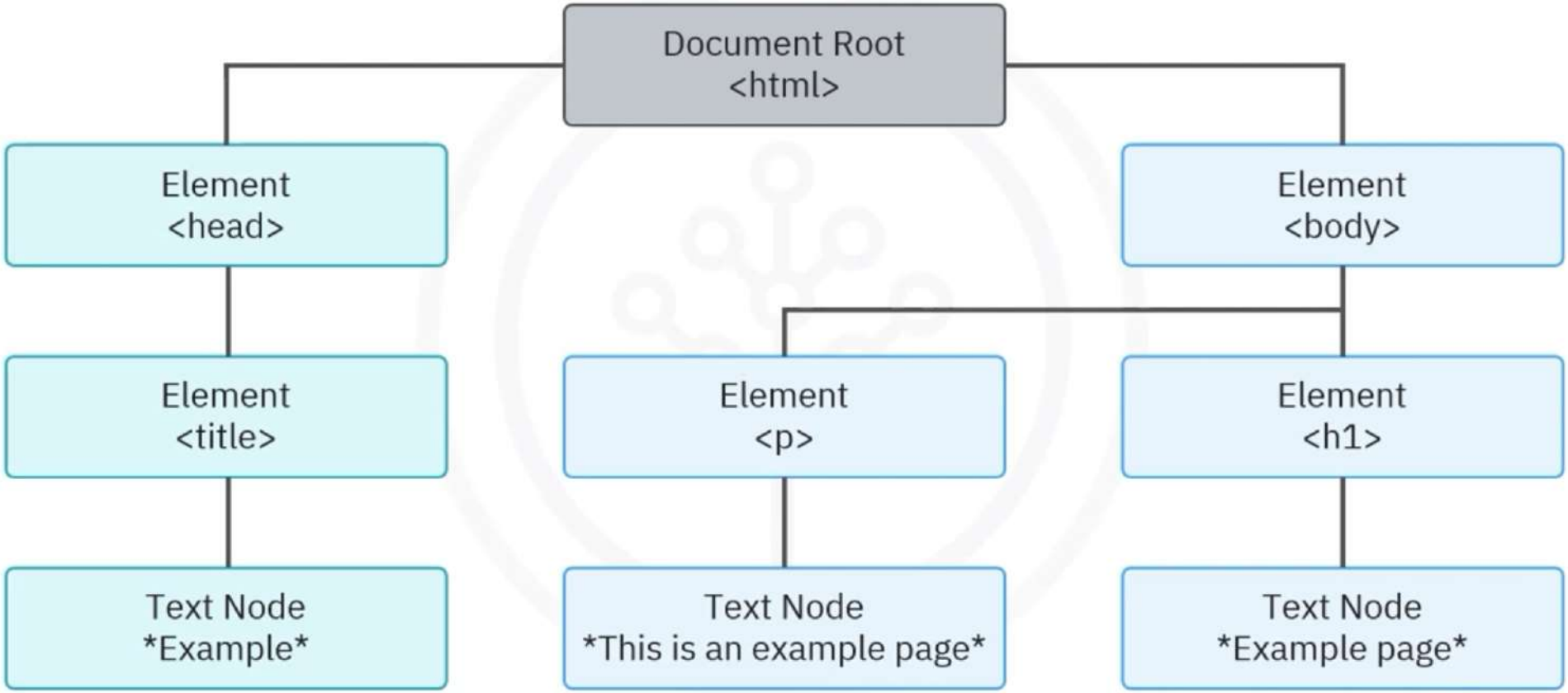
Define document object model (DOM) and its components

Describe the working of a virtual DOM in React

List the advantages of virtual DOM in React

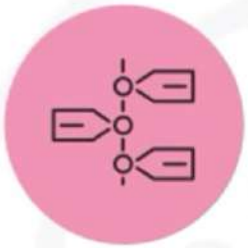
Compare normal and virtual DOMs

Document object model (DOM)



Components of DOM

Nodes
- Element
- Text
- Attribute

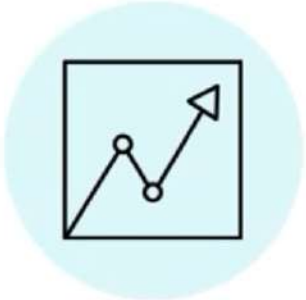


Elements
- Content

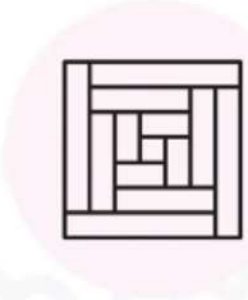
Attributes
- Properties
- Style
- Behavior

Events
- User interactions

Virtual DOM



Optimizes the performance of web applications



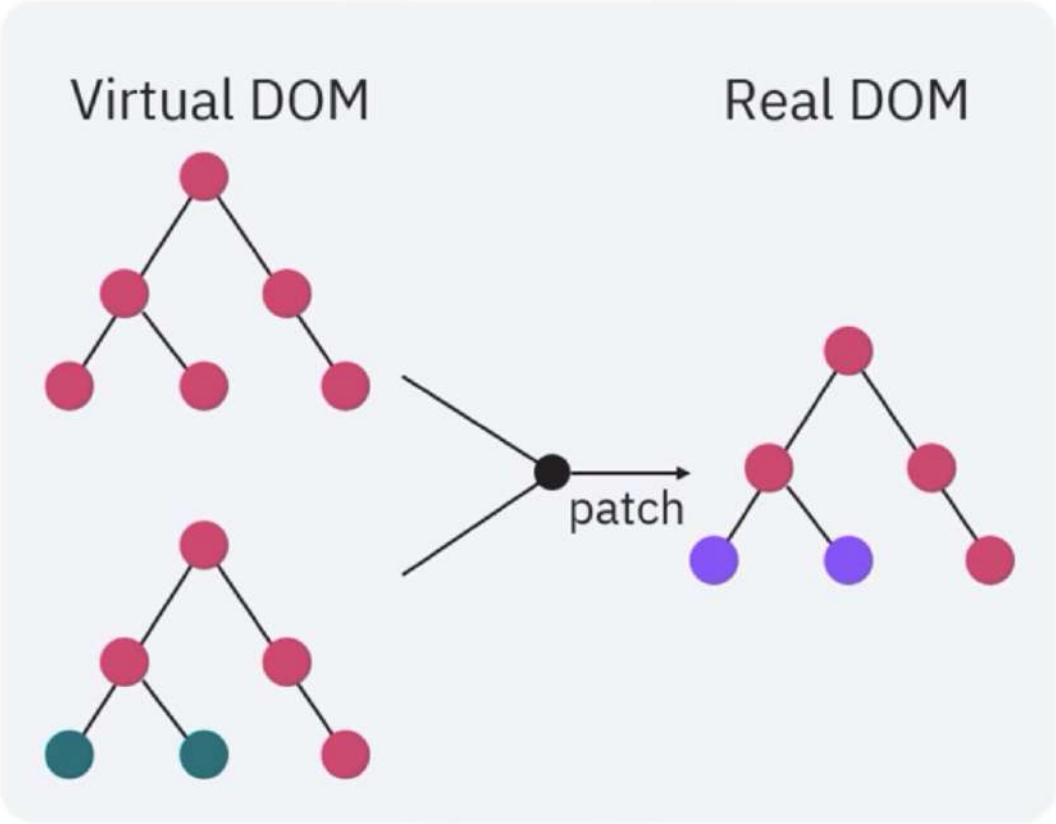
Abstraction of actual DOM in memory

Uses React's reconciliation process

How does the virtual DOM work?

Initial render Element
- Virtual model

Updates
- New virtual DOM



Diffing algorithm
- Identify changes

Patch
- Apply changes in batches

Why is virtual DOM important?

Speed enhancement

- Reduction in DOM changes

Simple development

- Concentrate on declarative code

Cross-platform compatibility

- Rendering on multiple platforms

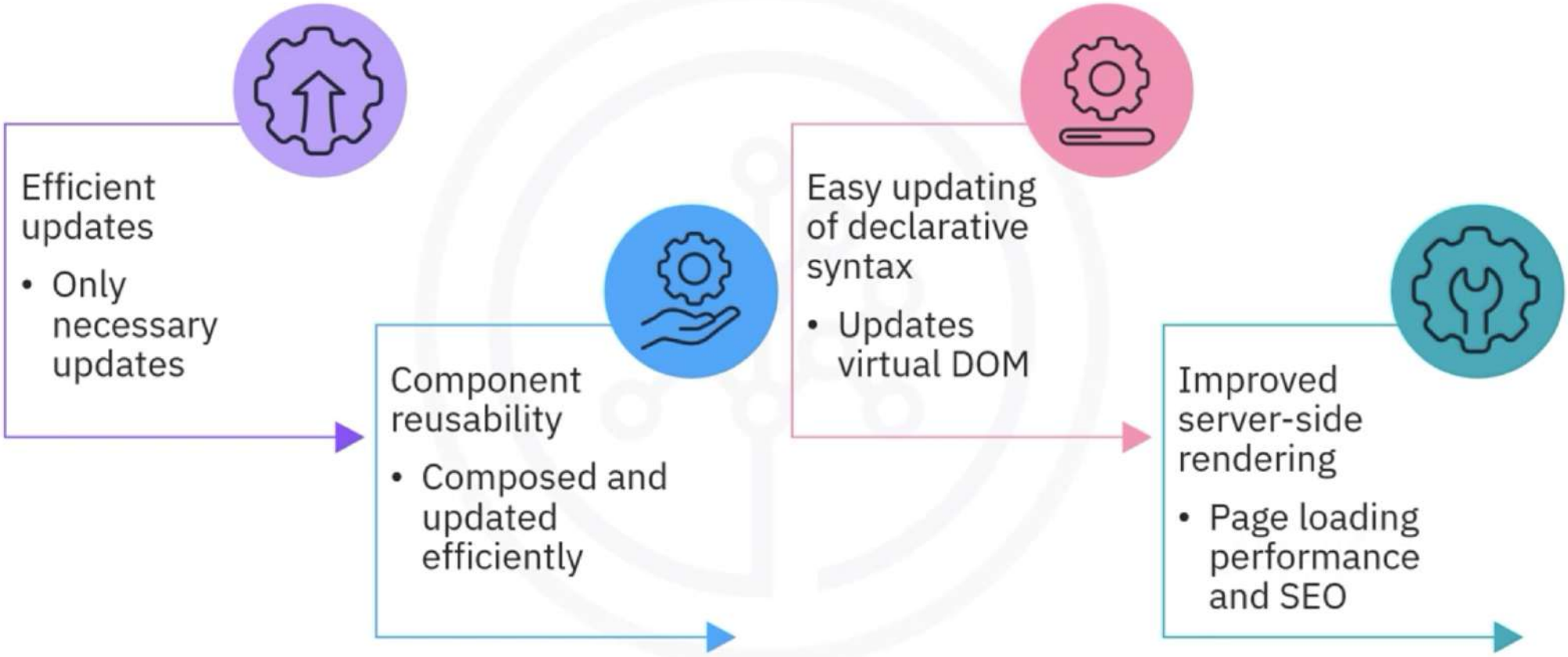
State management

- Effective updates and re-renders

Debugging and testing

- Examine virtual DOM version

Why virtual DOM in React?



Normal versus virtual DOMs

Normal DOM	Virtual DOM
Loading Process • Creates the DOM tree from the HTML code • Analyzes the HTML markup • Generates the DOM nodes • Displays the content on the screen • Rendering complex web pages can be expensive	**Loading Process** • Resides in memory • Does not update directly in real DOM • Creates a virtual DOM tree representation in memory • Based on initial state and properties

Comparison of normal and virtual DOMs

Normal DOM	Virtual DOM
Updates • Direct manipulation of the corresponding DOM nodes	**Updates** • Compares the current Virtual DOM with a new form of the Virtual DOM • Identifies the smallest set of changes • Applies the changes to real DOM • Makes only necessary changes • Triggers a reflow and repaint of the affected parts

React example

```
import React, { useState } from 'react'
import FirstChildComponent from './components/FirstChildComponent';
import SecondChildComponent from './components/SecondChildComponent';

const App = () => {
  console.log('Rendering App Component');
  return (
    <>
      <FirstChildComponent/>
      <SecondChildComponent/>
    </>
  )
}

export default App
```

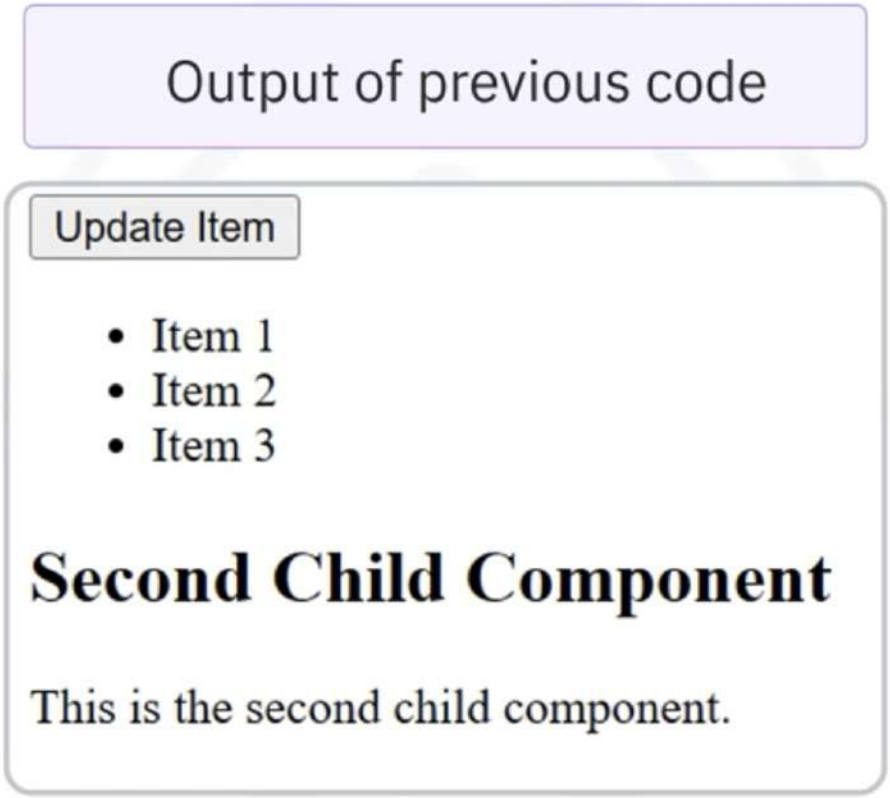
React example

```
import { useState } from "react";
const FirstChildComponent = () => {
  const [items, setItems] = useState(['Item 1', 'Item 2', 'Item 3']);
  const updateItem = () => {
    const updatedItems = [...items];
    updatedItems[0] = 'Updated Item 1';
    setItems(updatedItems);
  };
  console.log('Rendering First Child Component');
    return (
      <div>
        <button onClick={updateItem}>Update Item</button>
        <ul> { items.map((item, index) => (
          <li key={ index } > { item }</li> ))}
        </ul>
      </div>
    ); }; export default FirstChildComponent;
```

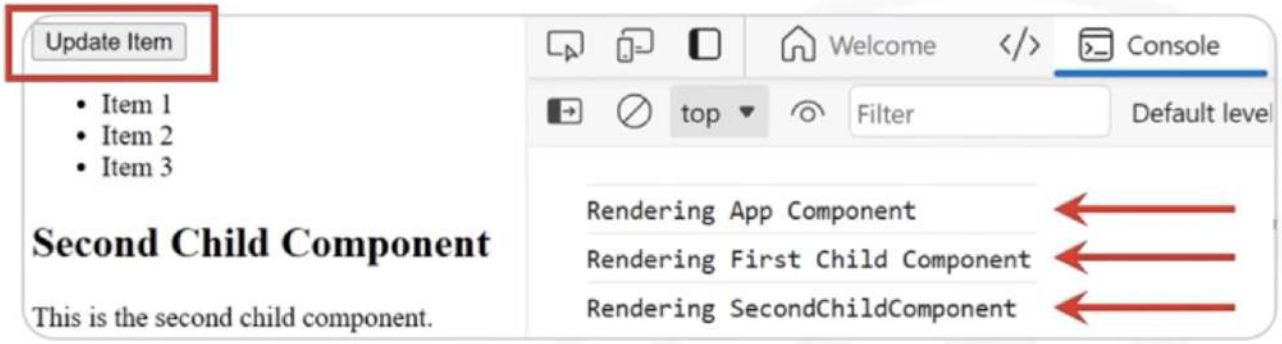
React example

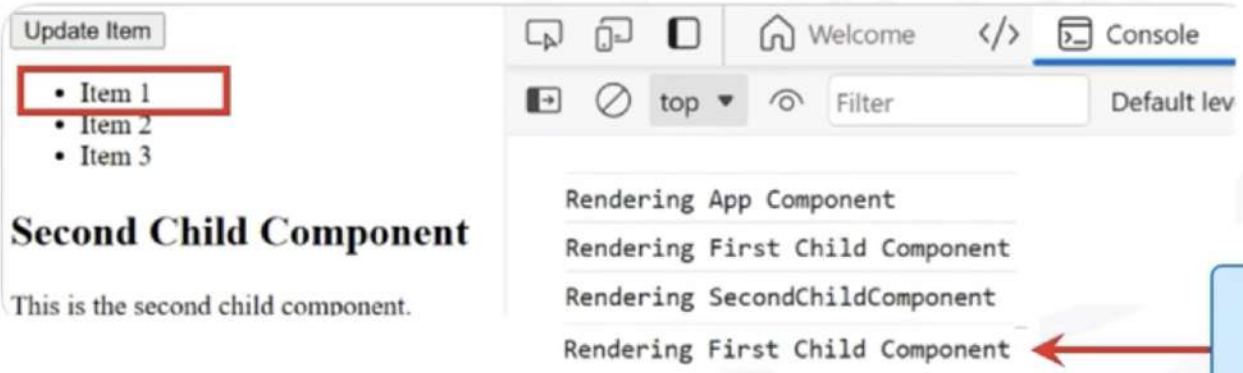
```
import React from 'react';

const SecondChildComponent = React.memo(() => {
  console.log('Rendering SecondChildComponent');
  return (
    <div>
      <h2>Second Child Component</h2>
      <p>This is the second child component.</p>
    </div>
  );
});

export default SecondChildComponent;
```

React example output



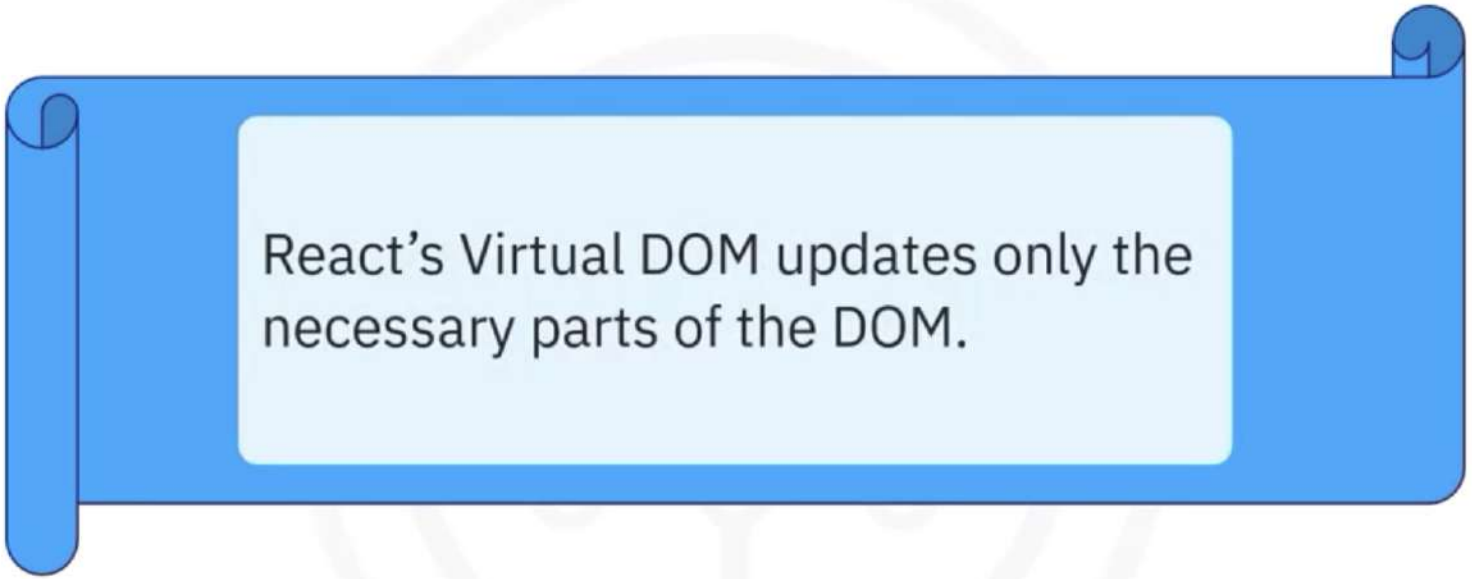
Console output





Only renders first item in `FirstChildComponent` list

Conclusion


React’s Virtual DOM updates only the necessary parts of the DOM.

Recap

In this video, you learned that:

- The DOM allows programs to access and manipulate web document's content, structure, and style dynamically
- Virtual DOM is an abstraction of the actual DOM implemented in memory and kept in sync with the real DOM by React’s reconciliation process
- React’s virtual DOM updates only the necessary parts of the DOM, improving the performance of the application, especially in scenarios where only specific components or elements need to be updated