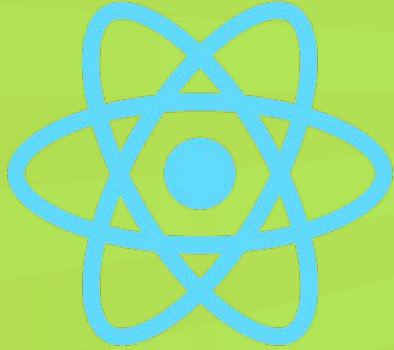




# Intro to React, Components, and JSX

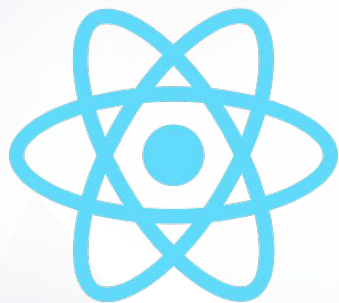
Web Development  
Lesson 19.1





React is one of the most powerful, in-demand front-end JavaScript libraries available today.





Just about every popular UI library that came after React borrows from it.

Libraries are like fads that come and go, but the ideas introduced by React appear to be here to stay.

# React

---

React is:



An open-source JavaScript library developed by Facebook for developing UIs.



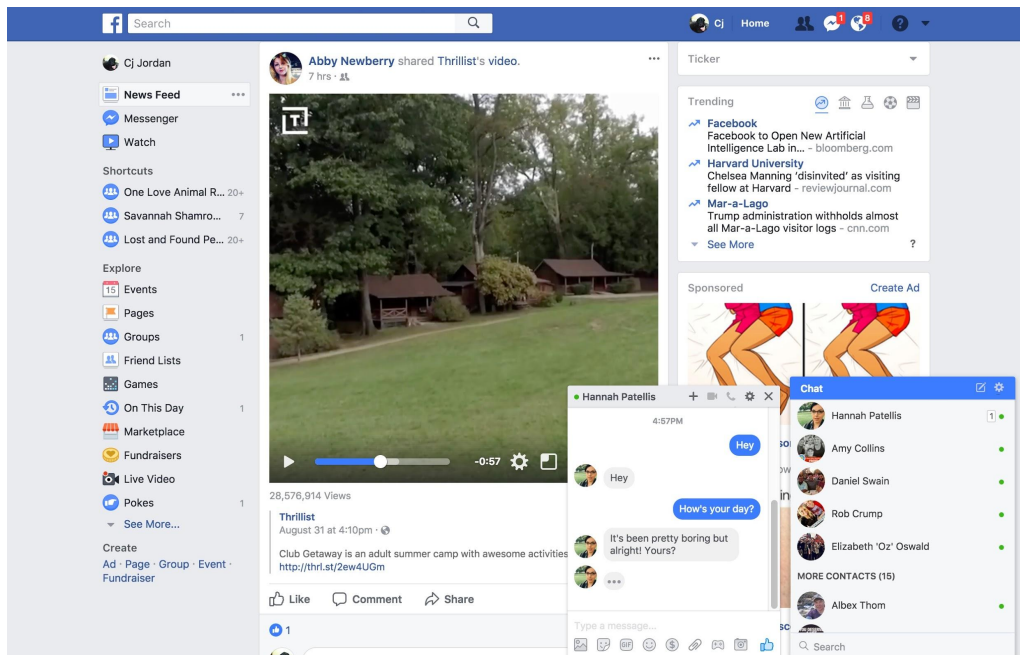
Developed to build large apps with rapidly changing data.



Component-based: UI elements are broken into self-contained components.

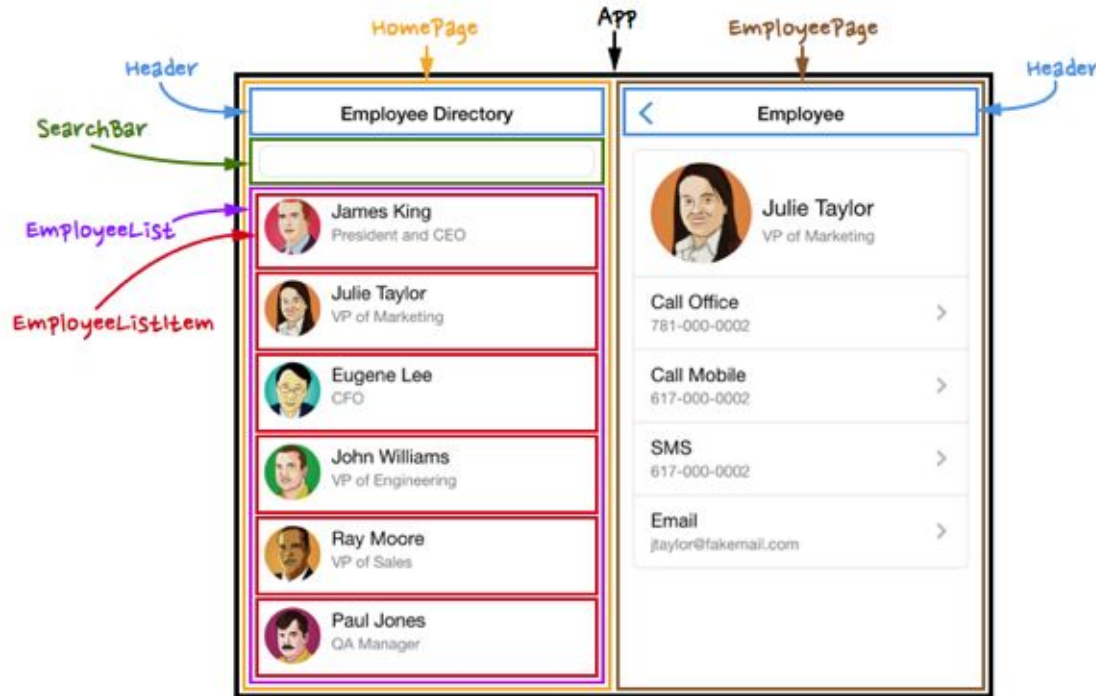
# Facebook's UI Complexities

Facebook buzzes with interactive options, live-updating data, and tightly interacting elements. This poses a challenge to simple DOM.



# The Concept of Components

Using React, UI elements are broken down into reusable components. Each sub-component behaves in a way that it is fully contained.



# The Power of Components

---

Why separate UI elements into components?



Logically decompose a UI into unique parts



Easily reuse these parts without re-coding



Easier to test



Helps you find bugs and saves time

# Data-Shifting Applications

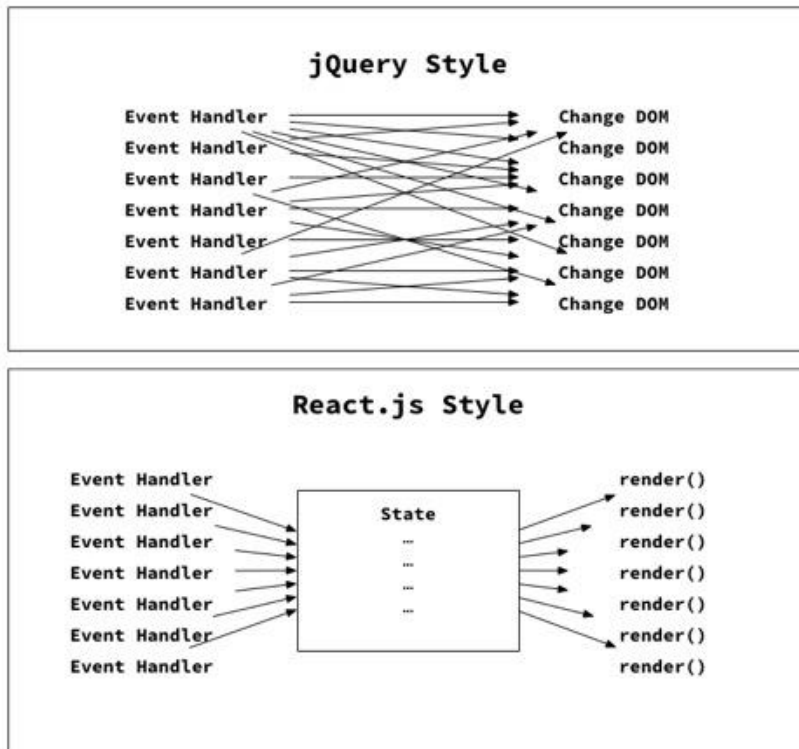


# How Is This Different Than jQuery?

In jQuery, the application's state and UI are updated independently of each other.

With React, whenever the application's state changes, the DOM updates to reflect it.

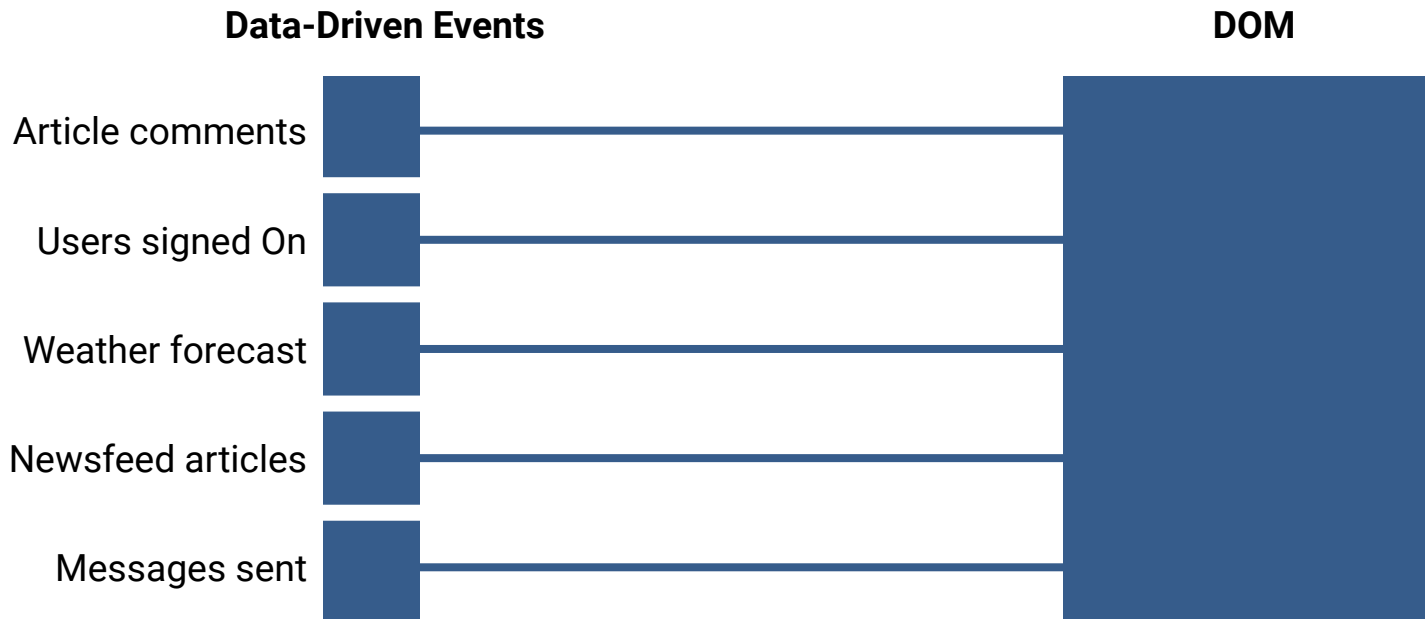
With React, the UI is a pure function of the application's state.



# Rapid Data Changes: Option #1 jQuery

---

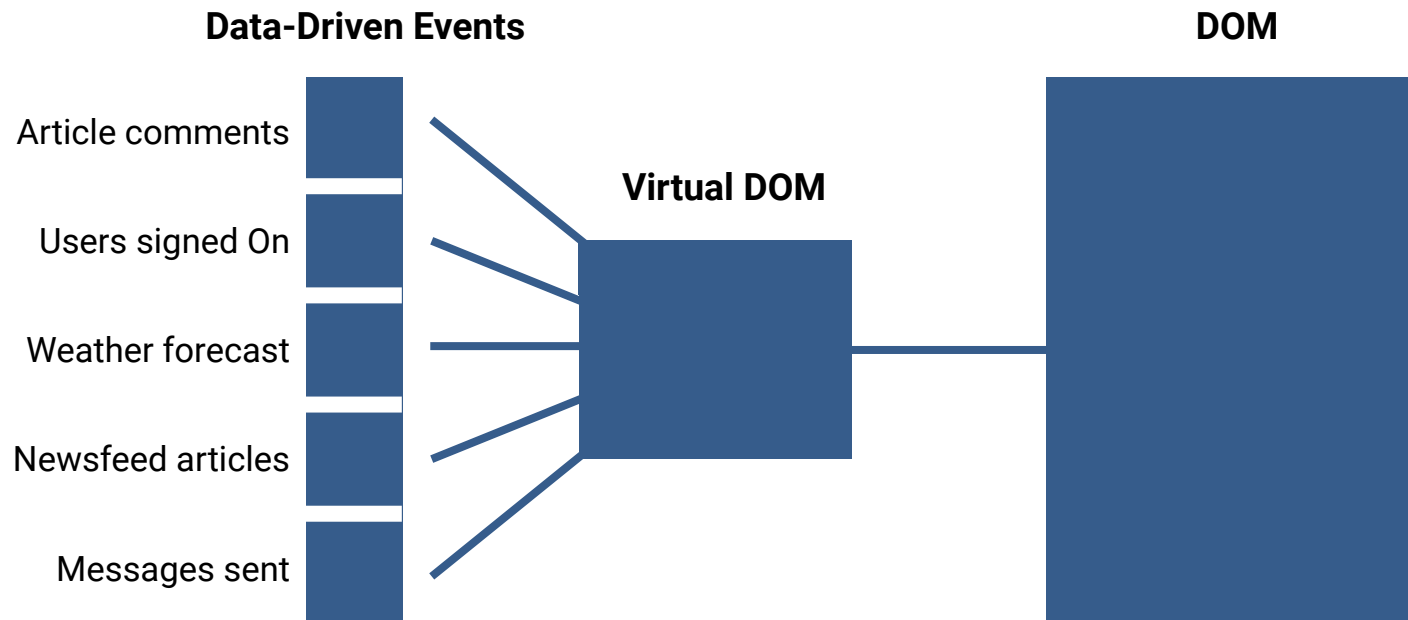
JavaScript is FAST, but whenever we update the DOM, the browser needs to recalculate the CSS, update the layout, and repaint the web page. This can be a slow process.



# Rapid Data Changes: Option #2 React

---

React's Virtual DOM serves as an intermediary and avoids unnecessary trips to the DOM.

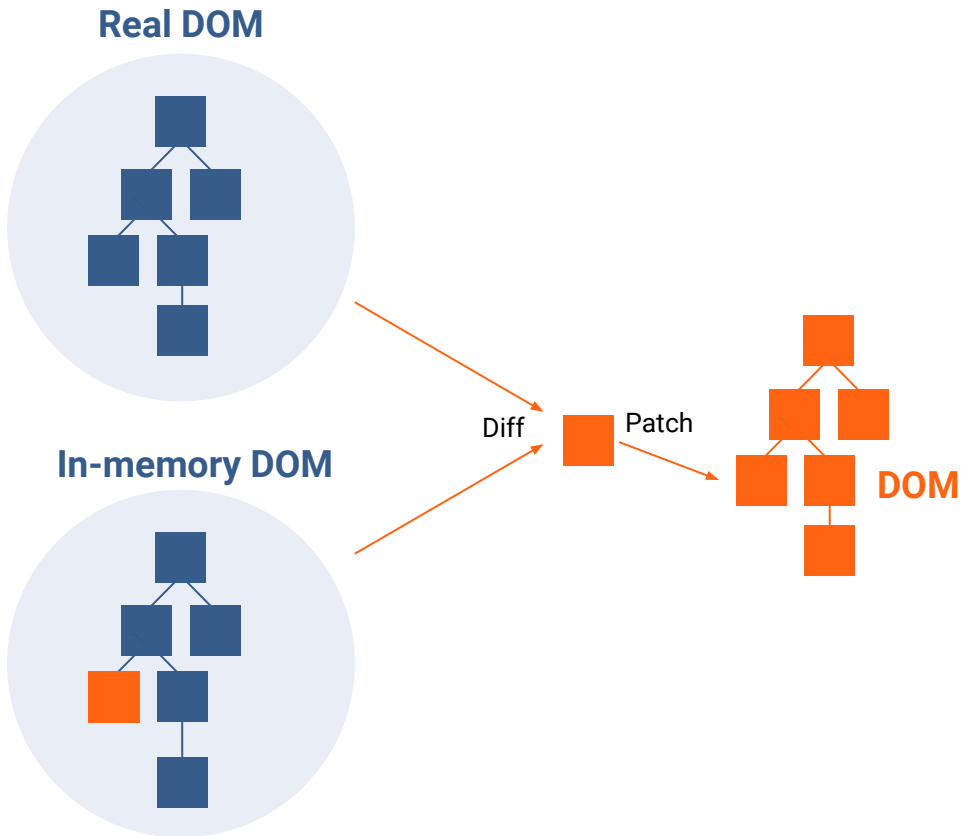


*What?*



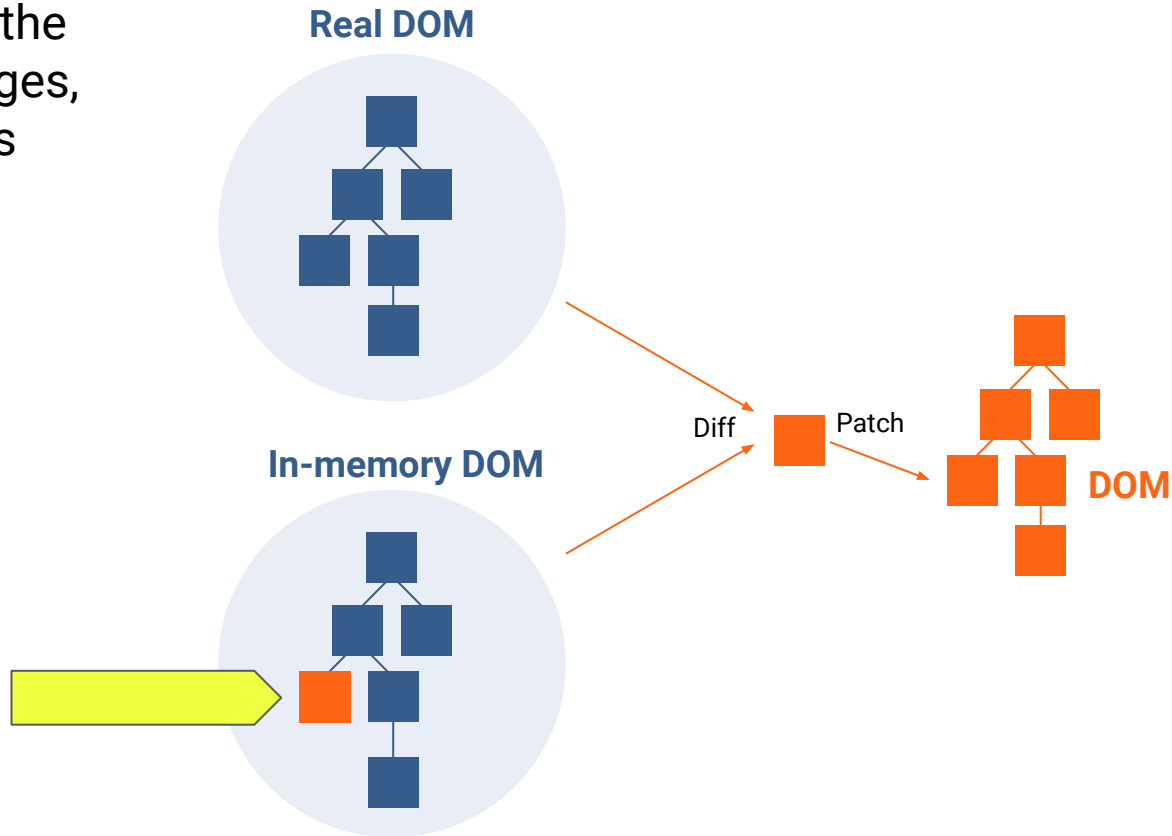
# Document Object Model (DOM)

A Virtual DOM is a JavaScript object that models the real DOM.



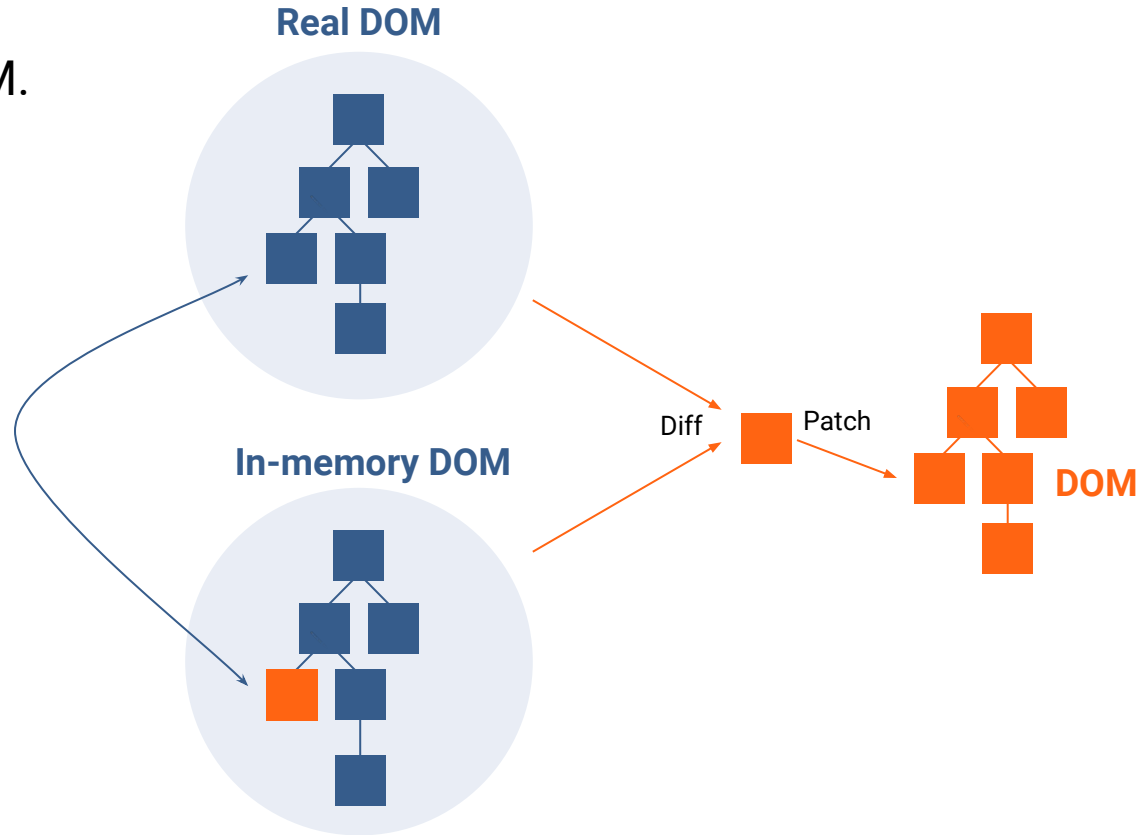
# Document Object Model (DOM)

Whenever some part of the application's state changes, the Virtual DOM receives the UI updates first.



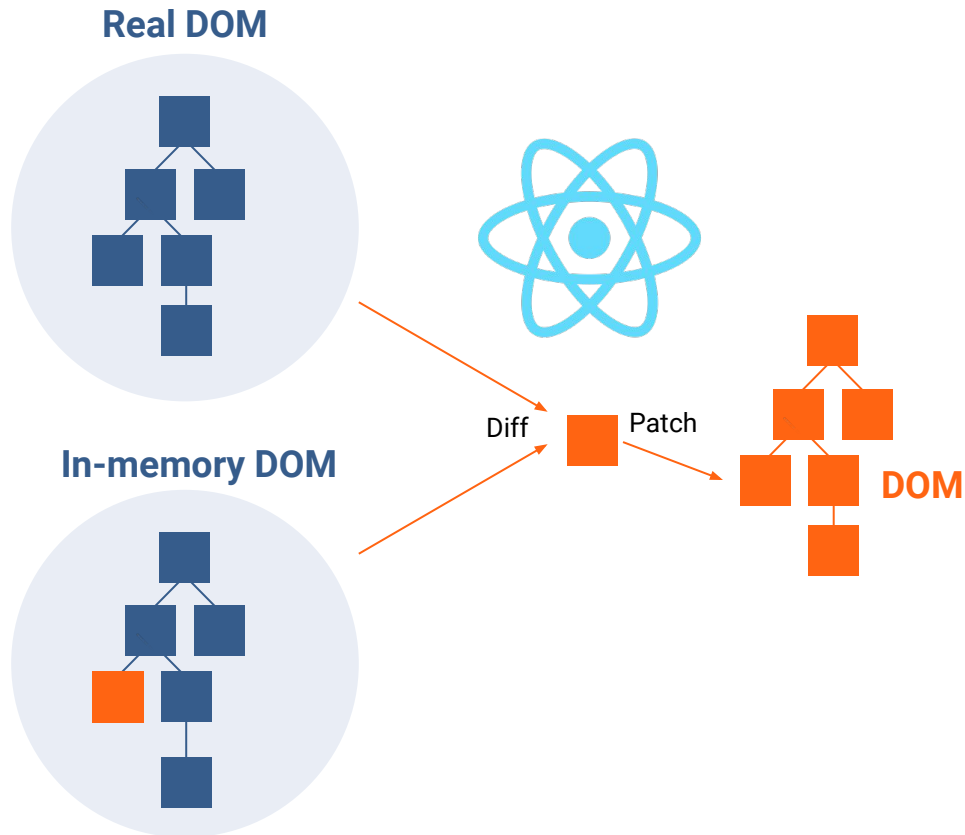
# Document Object Model (DOM)

Then the Virtual DOM is compared to the real DOM.



# Document Object Model (DOM)

React then aims to update with the smallest number of changes.





# React Pros and Cons

---

## Pros



- 1 Reusable components
- 2 UI updates in response to state change, reducing DOM manipulation code needed.
- 3 Can build applications on web, server, and native applications.
- 4 Easier to learn and more popular than other front-end JavaScript libraries and frameworks.

## Cons



- 1 React is a view library concerned with rendering user interfaces. You have to pull in other libraries to accomplish things like HTTP requests.
- 2 Can require more configuration than other libraries.

React Is Magical!



## Tools: webpack

---



[Webpack](#) lets you modularize front-end code the same way you do in Node with CommonJS modules (`require`, `module.exports`). Webpack also lets you apply various transformations on your assets via plugins.

# Babel Is a JavaScript Compiler

---



[Babel](#) lets you transpile next-generation JavaScript (ES6, ES7, ES8) into ES5 JavaScript that most browsers understand.

Can't I just use  
a **CDN** for  
all of this?







You could, but it'd  
be **SUPER** slow.



## **Activity:** Installation + Documentation Research

**Instructions sent out via Slack**

**Suggested Time:**  
15 minutes





**Time's Up!** Let's Review.



# <Time to Code>

