Software Development

# React

Module 12

# What Is **React**?

# React

React is:

01 One of the most powerful, in-demand front-end JavaScript libraries available today.

02 A JavaScript library that helps you create complex and responsive single-page applications.

03 Widely popular and well supported by the developer community.

04 Created by the developers at Facebook.

05 Makes code reusable and divides things into components.

# What Problem Does **React Solve**?

# What Problem Does React Solve?

React solves the following problems:

01

DOM operations are quite expensive in terms of performance, so React creates a **virtual DOM (VDOM)**.

02

The VDOM is a representation of the page structure in memory. It tracks what needs to be updated and only updates those specific things.

03

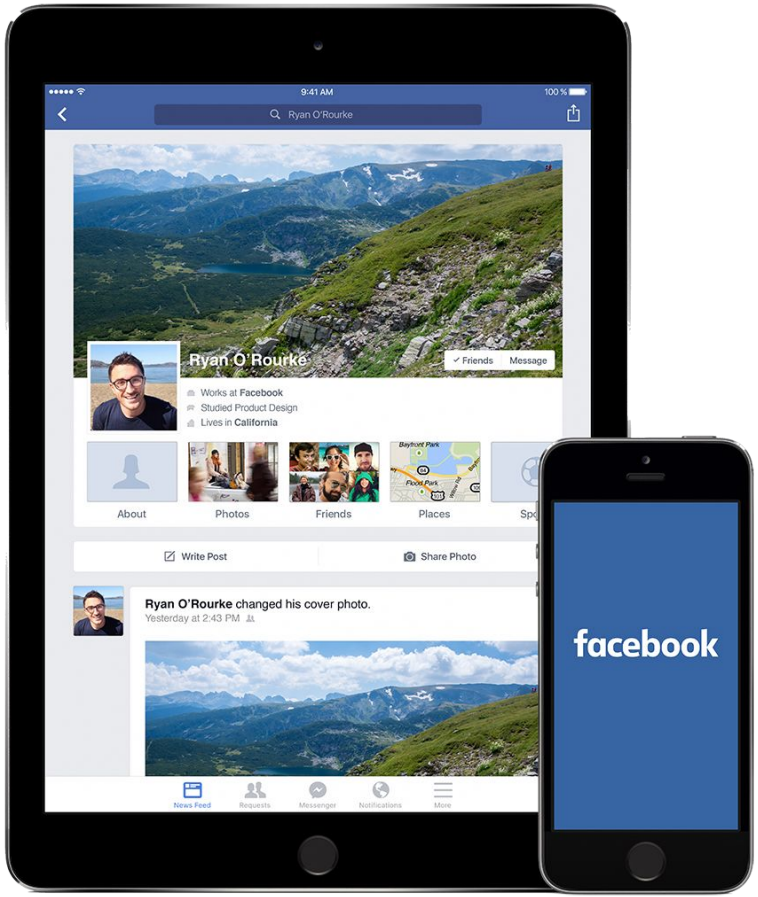
React is not opinionated like many other frameworks. It gives the developer the freedom to use Javascript the way they want to use it.

# Can You Give Me an **Example**?

# React Example: Facebook

Facebook's UI is a great example of React in action.

- Each section of the page is a component that itself has tons of real-time updates happening every second.
- The component design pattern allows Facebook to add a search bar and messenger to nearly every page that the user visits.



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

# What Are **Props**?

# React Props

Props are:

01

**Props** are a specialized type of parameter passed into a React component that help define attributes in the user interface, similar to DOM attributes.

02

Props allow data to be passed from a parent component to a child component.

03

Props is short for **properties** and refers to a normal JavaScript object that contains key-value pairs.

# Why **Separate** UI Components?

# Why Separate UI Components?

01 Logically decompose a UI into unique parts.

02 Easily reuse these parts without rewriting code.

03 Separate components are easier to test.

04 Helps isolate bugs, saving time.

# How Is This Different Than **Regular DOM Manipulation**?

# How Is This Different Than Regular DOM Manipulation?

01

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

02

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

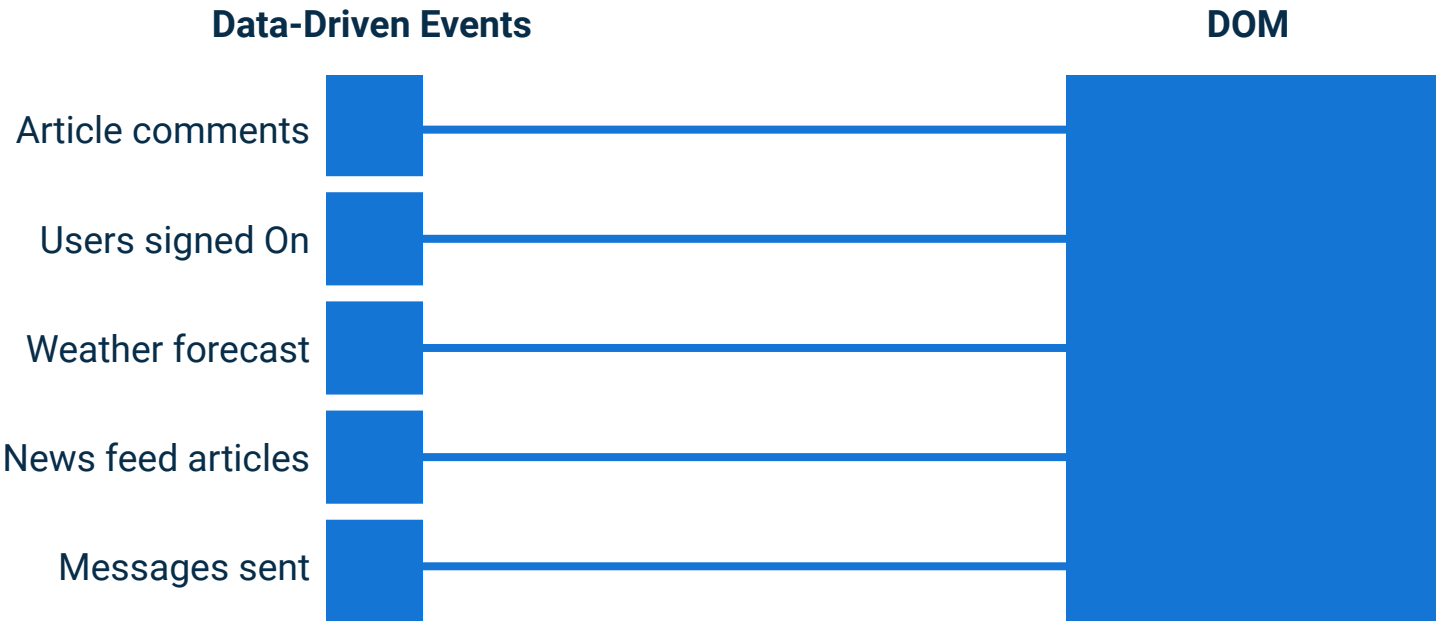
03

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

# How Do We Handle Constant **Data Changes**?

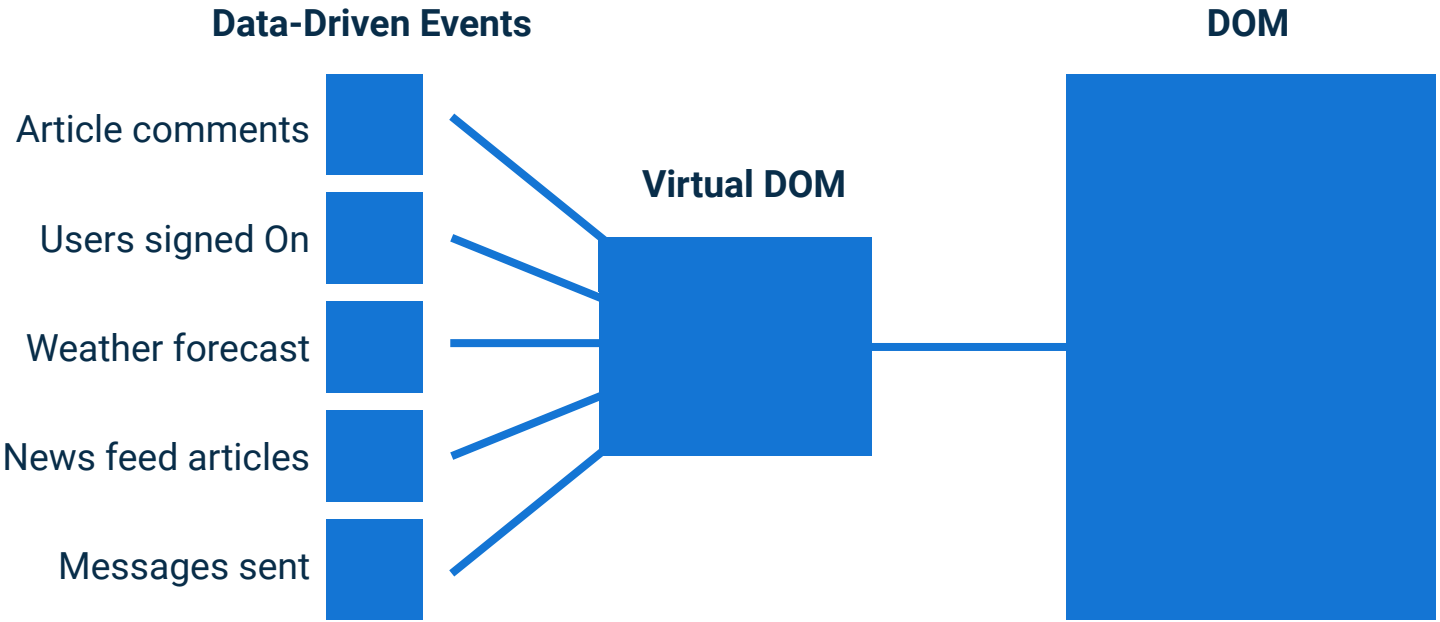
# Rapid Data Changes in Plain Vanilla Javascript

Rapid data changes in the plain vanilla Javascript requires the browser to recalculate the CSS, update the layout, and repaint the webpage. This can be a slow process.



# Rapid Data Changes in React

Reacting to Data Changes in Memory and avoids unnecessary trips to the DOM.



# What Is the **Virtual DOM**?

The virtual DOM is an ideal representation of the user interface kept in memory.

It is "synced" with the actual DOM through a process called **reconciliation**.

# What Is the Virtual DOM?

01

We tell React what state the UI ought to be in, and React ensures that the DOM matches the internal state.

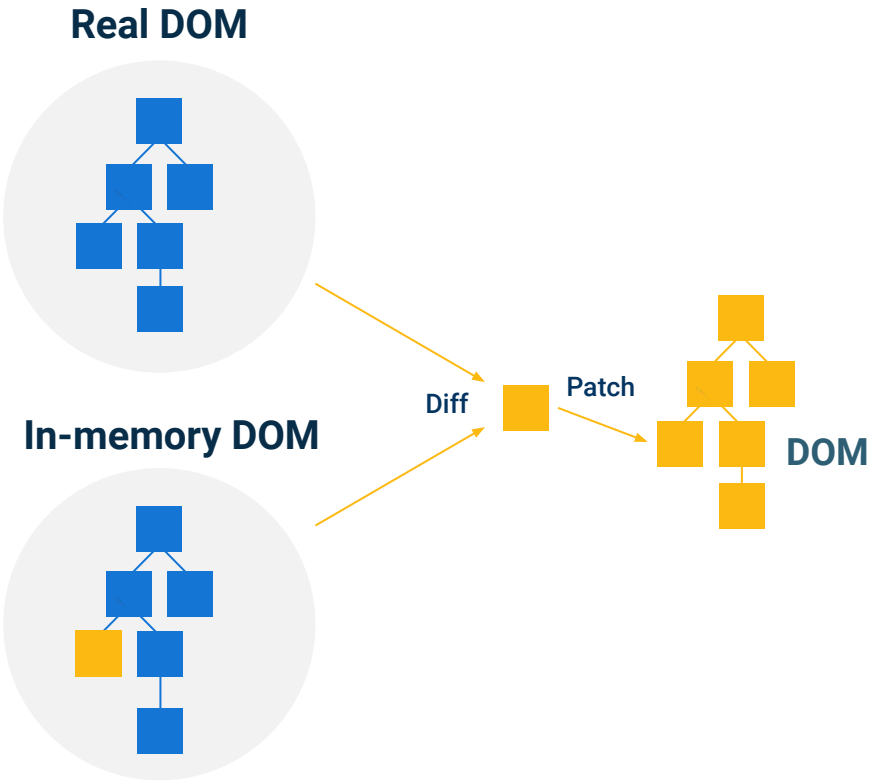
02

We isolate the attribute changes, event handling, and other DOM manipulation that we would otherwise use when building an app.

# Can We Get a **Visualization, Please**?

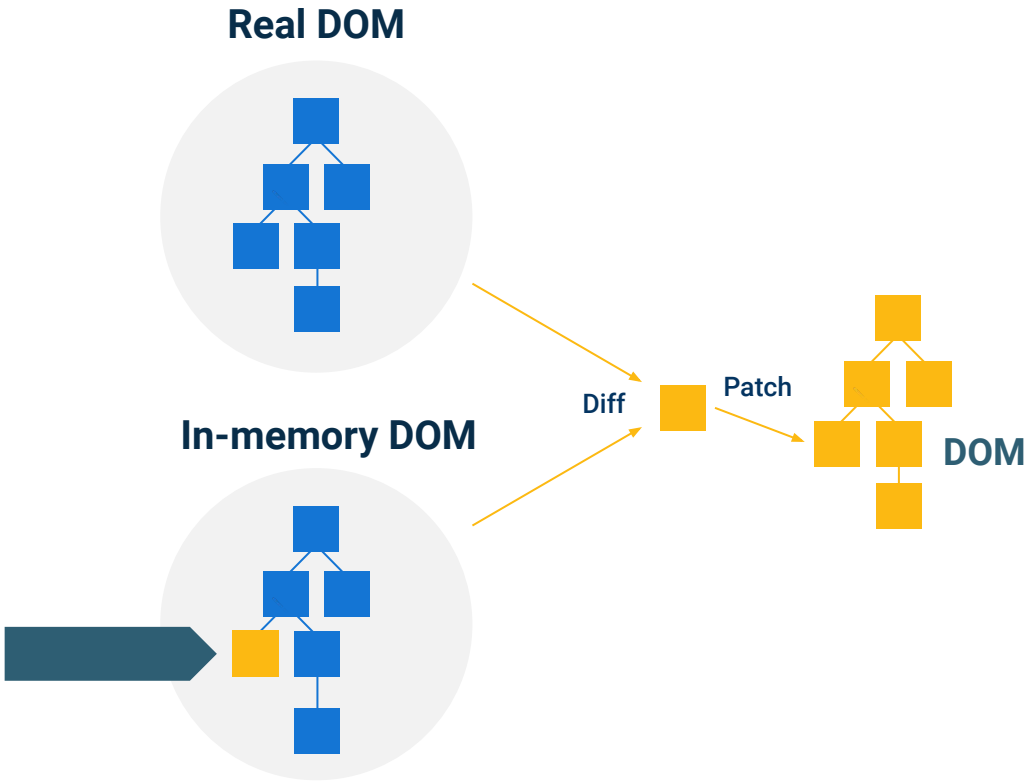
# Document Object Model (DOM)

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



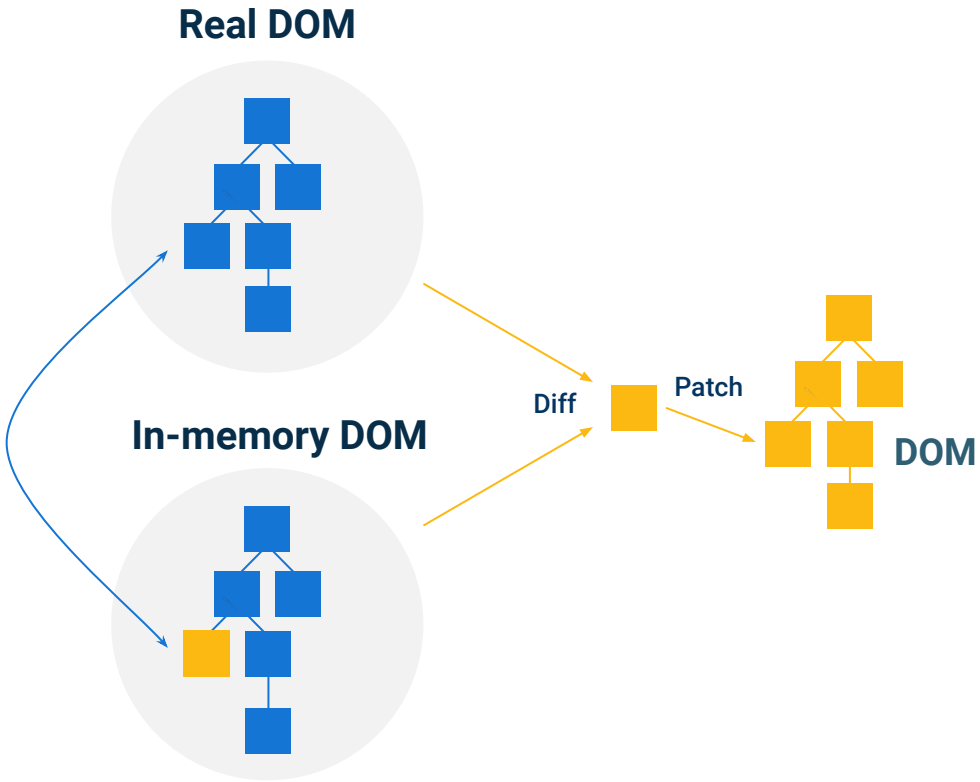
# Document Object Model (DOM)

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



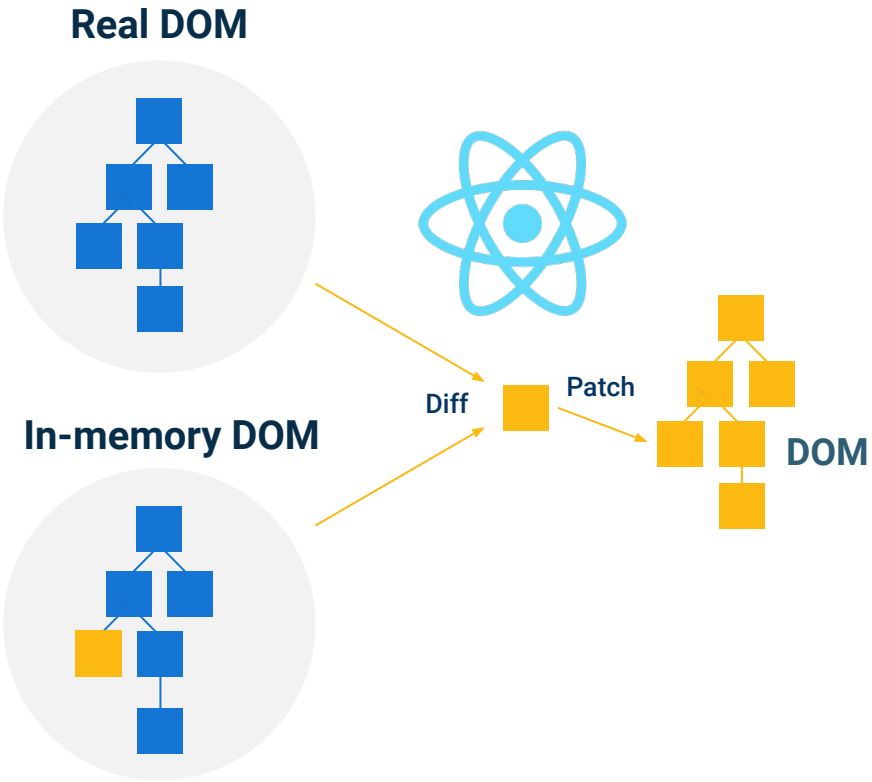
# Document Object Model (DOM)

Then the virtual DOM is compared to the real DOM.



# Document Object Model (DOM)

React then updates with the smallest number of changes.



# What Are the **Pros and Cons**?

# 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.

# What **Tooling** Is Needed?

# React Tooling

Vite

# What Is **Vite**?

**Vite** is a build tool that has gained popularity for its fast development server and optimized build process. Vite provides an optimized development experience out of the box, with native ES module support and rapid hot module replacement (HMR).

# What Are The **Benefits**?

- Fast Development Server
- Lightning-Fast Cold Start
- On-Demand Compilation
- ESBuild Integration
- Built-in Preprocessors
- Plugin Ecosystem
- Optimized Production Builds

# How Do We Learn **React**?

**React** was designed to help create performant single-page applications, but learning it can be daunting at first. Don’t worry—we will break it down into small digestible topics and take things one step at a time.

# How to Learn React

You can try the following strategies to learn React:

01 Read the official documentation and practice with the provided examples.

02 Reverse-engineer finished code to see how something was accomplished.

03 Build something from scratch.

04 Debug a broken React app.

05 And most importantly, ask questions!

# Instructor **Demonstration**

Mini-Project