Learning React

A JavaScript library for building user interfaces

Overview

- What is React?
- Why React?
- Components vs. Templates
- Virtual DOM
- Re-Rendering

What is React?

- A Library for creating user interfaces
- Not yet another JS Framework
- Renders your UI and responds to events
- The View in MVC

Why React?

Declarative, you will never touch the DOM (Document Object Model).

Component-Based, create your UI like Lego. Components are encapsulated that manages their own state.

React is just the UI, the rest is up to you.

Components vs. Template

- Separation of concerns reduces coupling and increases cohesion
- Components are the right way to sperate concerns
- Display Logic and Markup are highly cohesive, they both show the UI
- React Components use expressive power of programming language to build UIs (JSX)
- React Component, a highly cohesive building block for UIs loosely coupled with other components

Components vs. Template (Cont.)

- Single Page Applications (SPA)
- Offers more-native-app-like experience to the user.
- Built with frameworks/ libraries
  - React
  - Angular
  - Vue

Pros	Cons
• Most resources are loaded once. Only the data is transmitted back and forth. • Decreases load on the server. • Better UI/UX interactivity, it have the feel of native apps on mobile devices. • Static hosting and caching in a CDN.	• Slow initial render because heavy client framework or library are required to be loaded. • Non-SEO friendly, as content is loaded by AJAX. • It depends heavily on JavaScript. If a user disables JavaScript or the bundle failed to load, the app will fail to load.

Components vs. Template (Cont.)

- Multiple-Page Applications (MPA)
- MPA usually built with a templating engine
  - Pug formerly known as Jade
  - Handlebars
  - Template7
  - Etc.

Pros	Cons
• Fast initial render of the app, as the content is rendered by the server before sending the markup. • SEO friendly, as content is preloaded.	• Increased load on the server, because for each request the server will generate new markup for that request. • Latency due to page requests/reloads. • Low UI/UX interactivity.

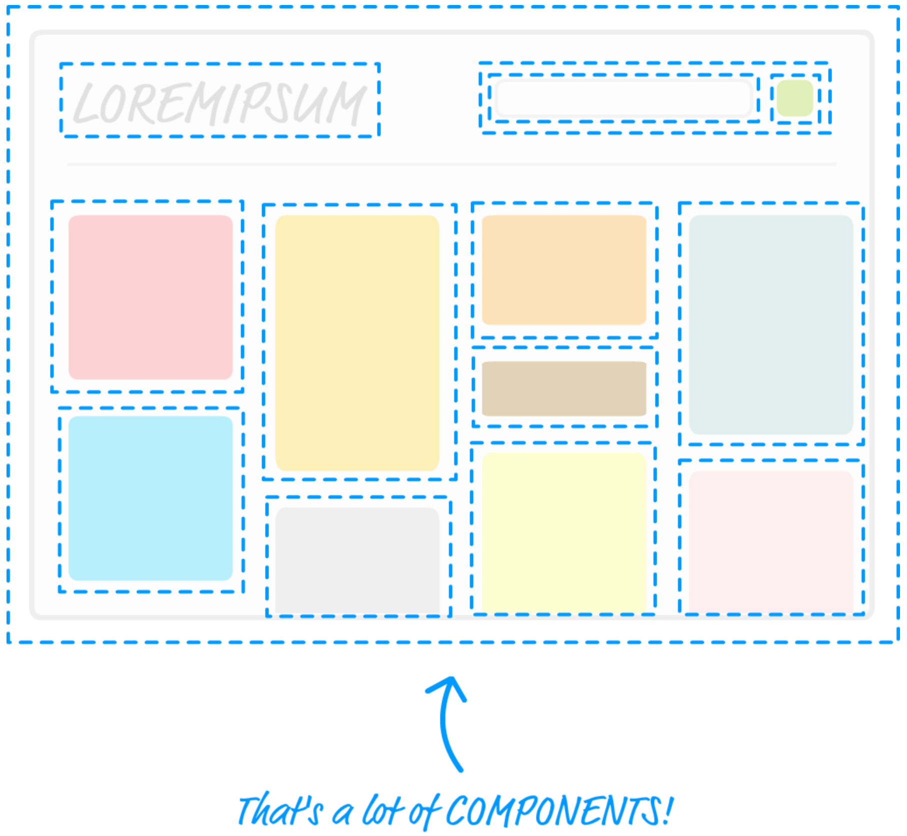
More Components

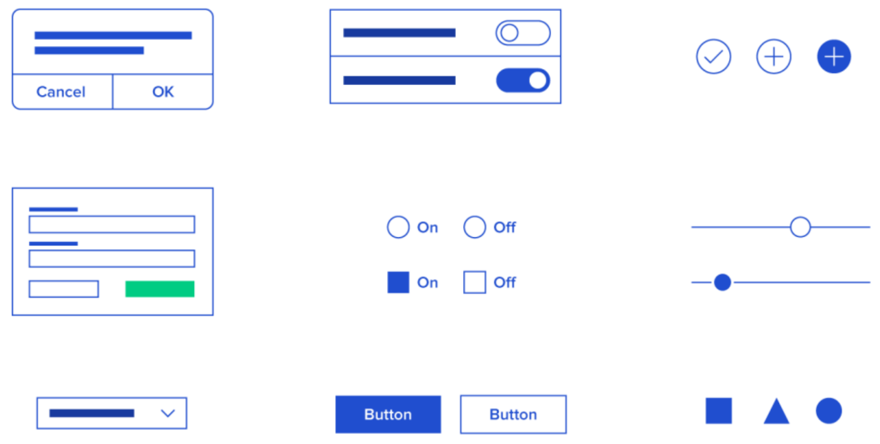
- Components are Reusable, Composable and Unit Testable
- Build many simple components that does one thing well
- Compose them into a bigger functional unit
- Allows to structure the application better with more flexibility

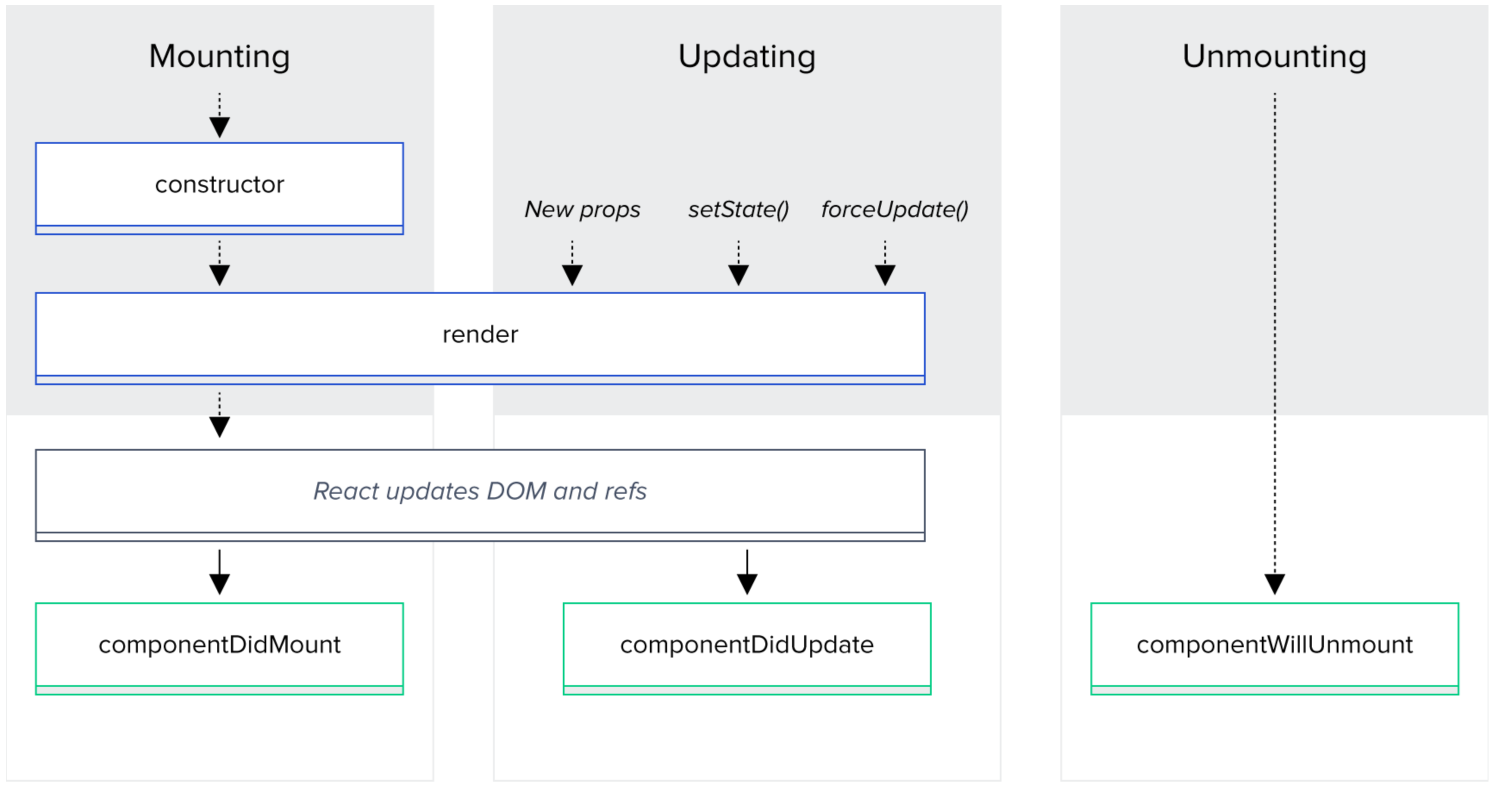
More Components (Cont.)

- State has data you own, Props has data you borrow
- When the state changes, React re-renders the entire component
- No data binding or model dirty checking
- No more explicit DOM (Document Object Model) operations, everything is declarative

Components Illustration





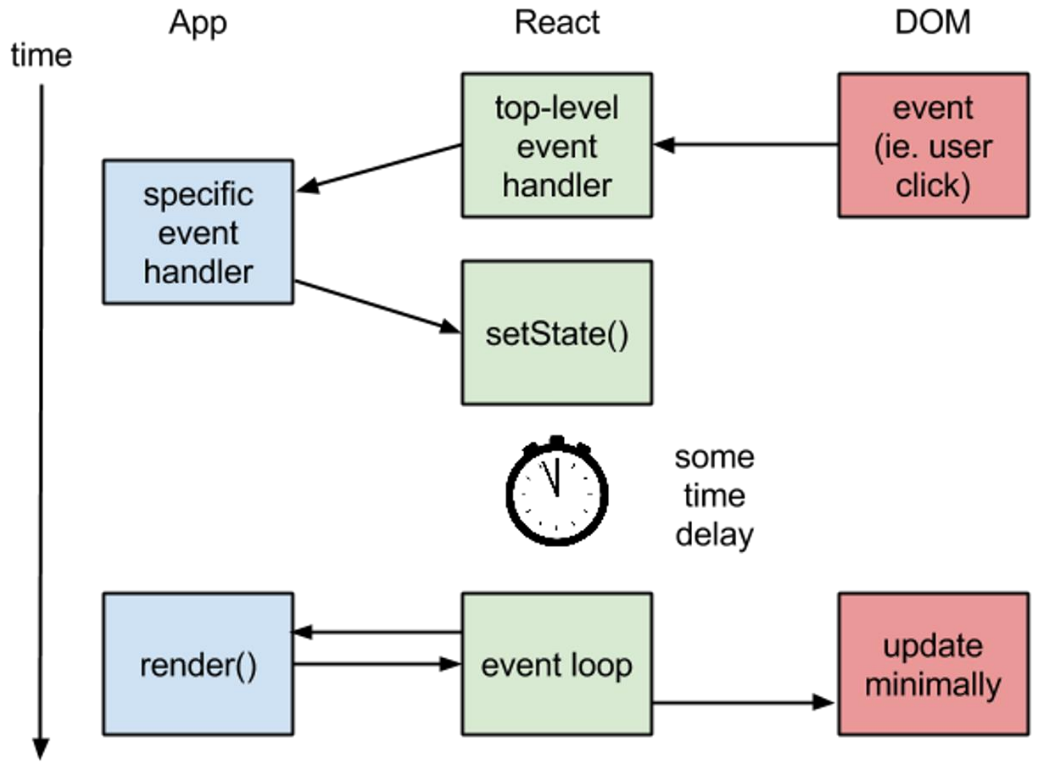


More Components (Cont.)

- JSX is a JavaScript syntax extension that looks like XML
- Stateful Components have Lifecycle and State
- Stateless Components do not have Lifecycle neither State

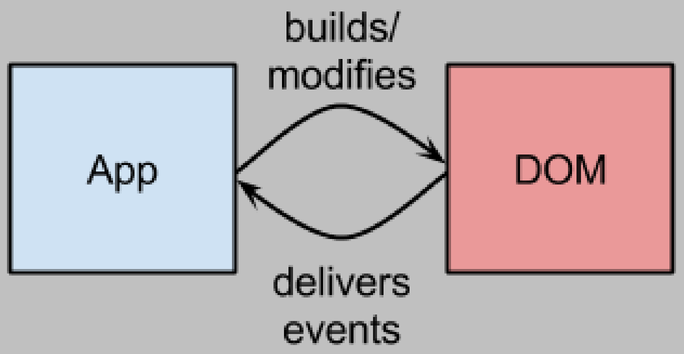
Virtual DOM

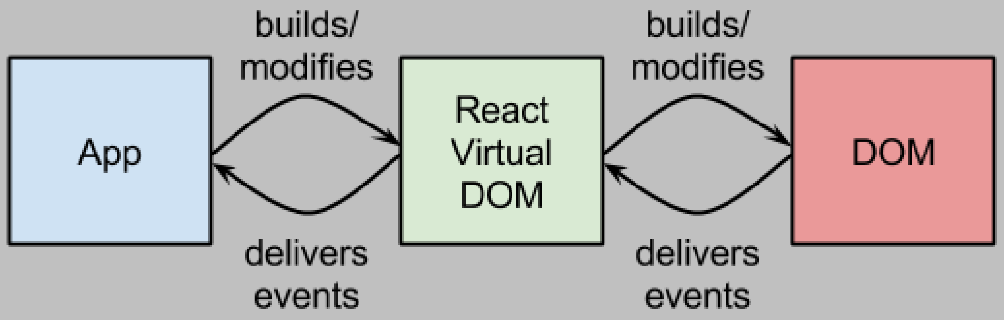
Data flow when a user event results in a DOM update



React vs Traditional Web Apps

- Makes re-rendering on every change faster
- Computes minimal DOM operations
- Batched reads and writes for optimal DOM performance





On Every Update…

React builds a new virtual DOM **subtree**

Diff it with the previous subtree

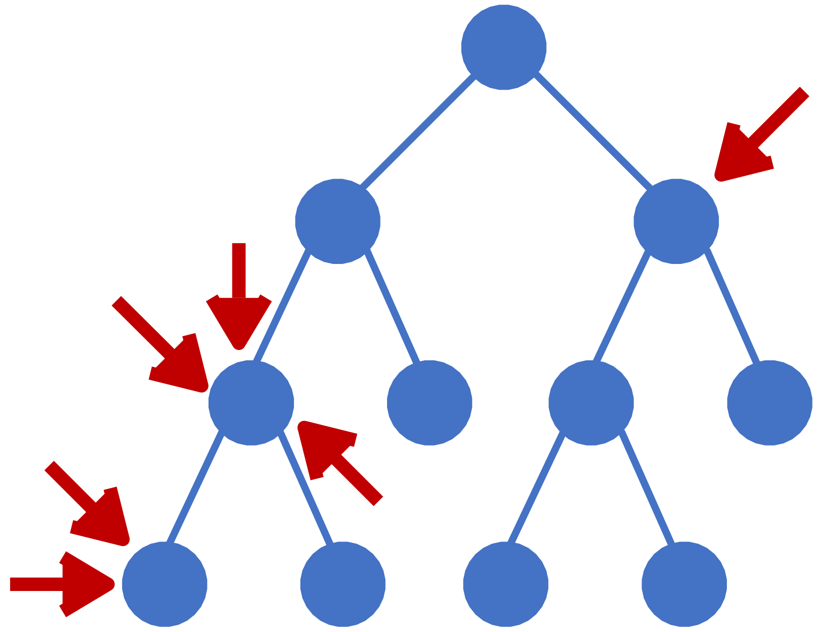
Computes the minimal set of DOM mutations and puts them in a queue

Batch execution of all mutations

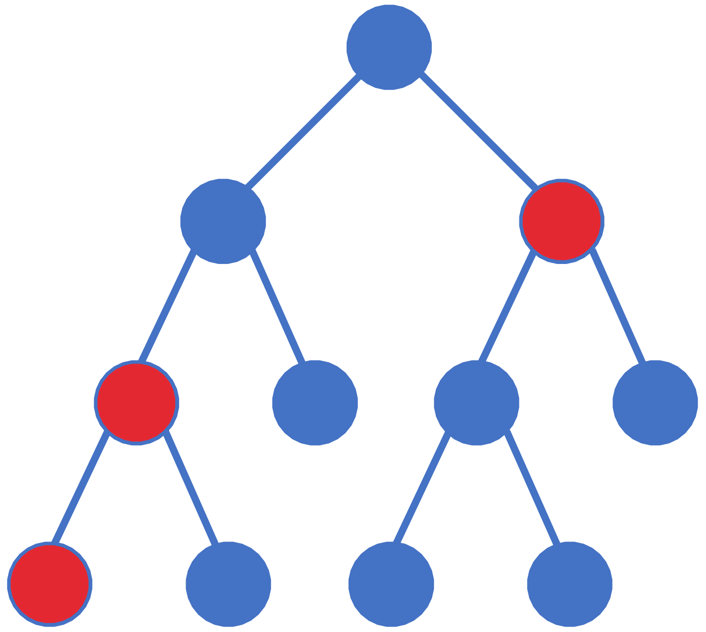
Re-Rendering Process

Batching DOM operations

Setting State

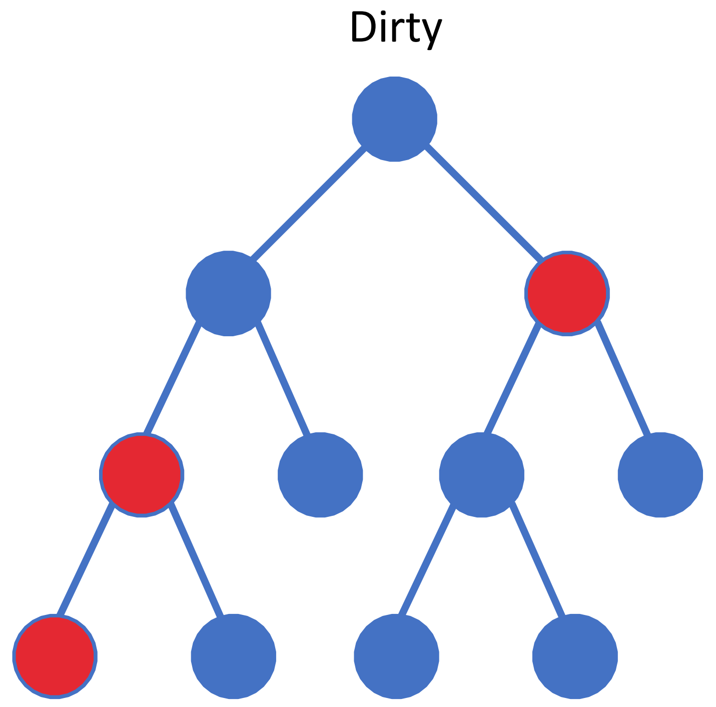


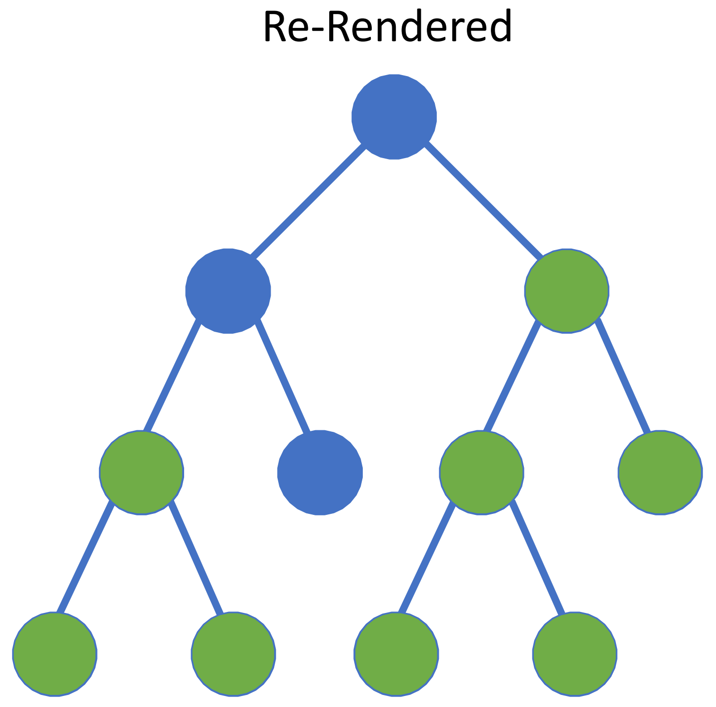
Dirty



Re-Rendering Process

Streaming batched DOM operations




Re-Rendered