What's React

Javascript developers hate this one framework

Justin
May 18, 2020

The TL;DR

React is a philosophy and framework for building interactive web (and native) applications that are built on reusable components.

- Web apps are increasingly using **client side rendering instead of server side** (scary words! we'll explain them)
- Web developers use React to build **dynamic web applications** like Twitter
- React's philosophy emphasizes reusable **components** so you don't rewrite code
- The **must-know basics** and pieces of the React ecosystem: props, JSX, state, and React Native vs. React JS

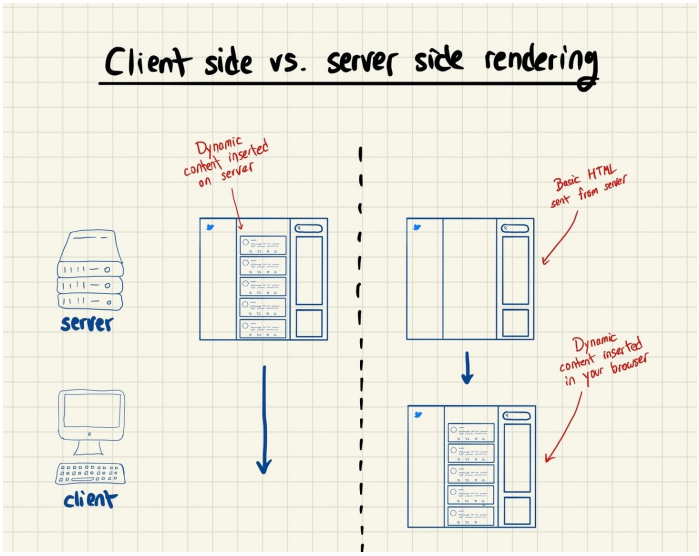
Unless you live under a rock (which, given San Francisco rents, is becoming a legitimate housing option), you've probably heard about React. It's at the center of *a lot* of hot topics right now, like "are frontend engineers really engineers" (hint: yes, yes they are) and my personal favorite, "what's the most complicated way I could possibly build this one page website." So what is React, and why should you care?

Client side rendering vs. server side rendering

> 🔮 **Dependencies** 🔮
>
> To get the most out of this post, you'll want to understand how a web app works and what dynamic content looks like.
>
> 🔮 **Dependencies** 🔮

Hopefully you've been reading some Technically in your free time, and you're familiar with the client-server model. It's how the whole web works: companies host their websites and apps on fancy, powerful servers, and you download files from those servers to look at in your browser. When these files are *dynamic*, their content changes based on who's logged in, what's going on, what screen you're on, and stuff like that. This section is about *how* that actually happens.

When an app or website needs to send you dynamic content, there are (surprise!) two places they can *create* that content: on the server side, or on the client side. The server side means they populate and configure the page *before* they send it to you; the client side means they send you the basic HTML, and the dynamic logic happens *in your browser*.



In practice, the difference is in where the backend logic happens: all on the server, or mostly in the browser calling APIs on the server.

The benefit of doing things server side – in a lot of cases, it's faster. That's because files only need to make one trip – from the server to you – as opposed to Javascript in your browser hitting multiple APIs on the server over time to load one page.

The benefit of doing things client side – you don't need to reload the same HTML and other boilerplate stuff every time you load a page; all that needs to change is the dynamic parts of whatever you're loading.

Traditionally, server side rendering was the dominant form of building web applications. But as Javascript has gotten better and web apps have gotten more complicated, client side rendering is getting more popular, to the point where it's now pretty much the default for most tech startups building their stuff from scratch. React is the poster child of that movement.

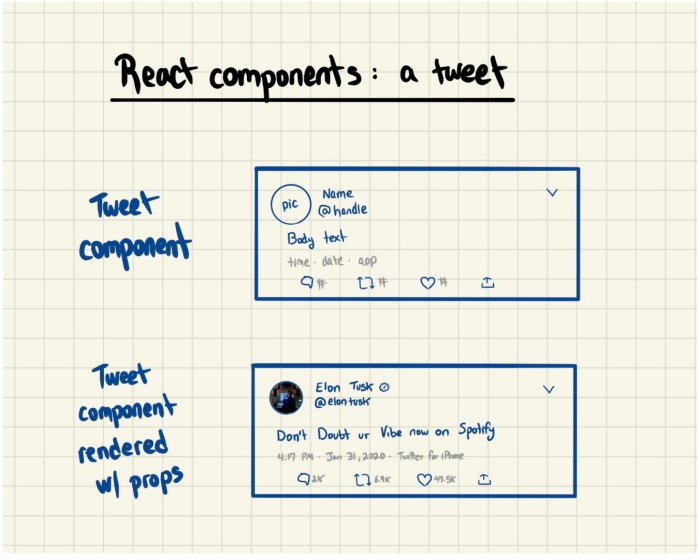
(Nowadays, things aren't completely black and white: a new class of applications is combining server side and client side rendering into hybrid models that take advantage of the benefits of both.)

React's philosophy: components

React is a *library* for building these kinds of interactive apps. A few engineers at Facebook built it almost 10 years ago – it was present in the 2011 News Feed – and it slowly got more popular over time. A framework is just a bunch of code that helps you write other code, so let's explore exactly what React is about and what it helps you do.

The core philosophy of React is **all about components**. If you take a look at popular web apps that you use – Twitter, Google Drive, or even Wikipedia – there's a lot of *repetition* going on. Each tweet that you see in the web app looks pretty much the same; it's a box of a certain width and color with some text in it and a border. All Google Drive files look pretty similar when you load your homepage, too. In React, these are components, and they're built to repeat, a lot.

Instead of writing the same HTML for every single file icon in your Google Drive, the Google engineering team might build a *React Component* called a file. That file has a basic HTML structure (box, border, shadow, etc.), but the rest is dynamic: it will show the green icon if it's a spreadsheet, and the blue icon if it's a doc. The name of the file, as well as the time it was last updated, is also dynamic (it changes from component to component).



In your typical web app, you might have anywhere from a few to several hundred components. The important thing to understand here is that this focus on components is a *fundamental pattern shift* – people did not always build their

sites this way. Part of React's popularity is tied to the paradigm shift, not just how useful it is as a framework.

One last important thing to note: developers also build server side rendered apps that use React. One of the most popular frameworks for React is Next.js, which helps devs build apps like that.

React 101 for non-coders

Let's run through a few key parts of the React ecosystem so you can get a better sense of what developers are doing. If some of these don't click the first time, that's totally OK; it's more important to get the broader ideas at play.

Props

Each component is a combination of things that don't change (sometimes color, size, etc.) and things that do change (text, images, etc.). When you write React components, you pass the things that do change – the dynamic elements – to the component through *props*. The word might be short for properties. I don't know.

JSX

HTML and Javascript are separate languages with separate goals. But one of React's innovations was to sort of combine them: JSX is a React syntax extension that lets you write HTML *within* Javascript so you define what your components should look like all in one place.

State

Sometimes your React components need to remember things. If you're building a little timer, you add and display a second every second; so you need to remember what second we're at. React has special built in functionality for storing data like that, and there's even an entire framework for managing React state.

> ☕ **Dev Culture** ☕
>
> React might be popular, but it's not perfect. It has a bit of a reputation in the developer world of being difficult to get started with, and kind of "overkill" for simpler projects.
>
> 
>
>
> ☕ **Dev Culture** ☕

React Native

The main "part" of React is a Javascript framework used for building *web* applications. But there is a version of React that you can use to write *native apps* – applications that run on your laptop or mobile phone, not on the web. React Native is also built and maintained by the Facebook team.

"React" in conversation

"Just take it from our component library."

We have a bunch of components that we've built and use across the app: instead of creating something from scratch, use one of those components.

"You could turn it into a component if you want, but I'm not sure how reusable it's going to be."

It's only worth making a bunch of HTML into a component if you're going to reuse it across the site; if it's a one off thing, don't bother.

"We moved our mobile app over to React Native because our developers were more familiar with it."

We were building our iOS app in Swift, but ended up moving it over to React

We were building our iOS app in Swift, but ended up moving it over to React native because our developers already knew React.js.

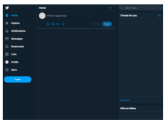
Terms and concepts covered

```
Client side rendering
Server side rendering
React
Components
Props
JSX
State
```

Further reading

- React is the most popular client side rendering framework, but it's not the only one; Google pioneered the idea with Angular in 2010, and Vue.js is getting some traction too
- Redux is a popular library for managing application state, and there's a special React version called React Redux

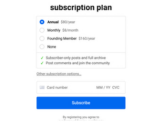
Justin Sep 13, 2021 28 5



What's Headless E-Commerce?

We may be running out of names

Justin Nov 2, 2021 20 7



I built a (basic) Substack clone in a month

This was probably a waste of my time

Justin Sep 2, 2020 50 4

See all ›