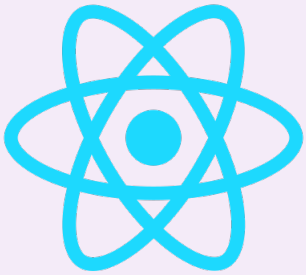


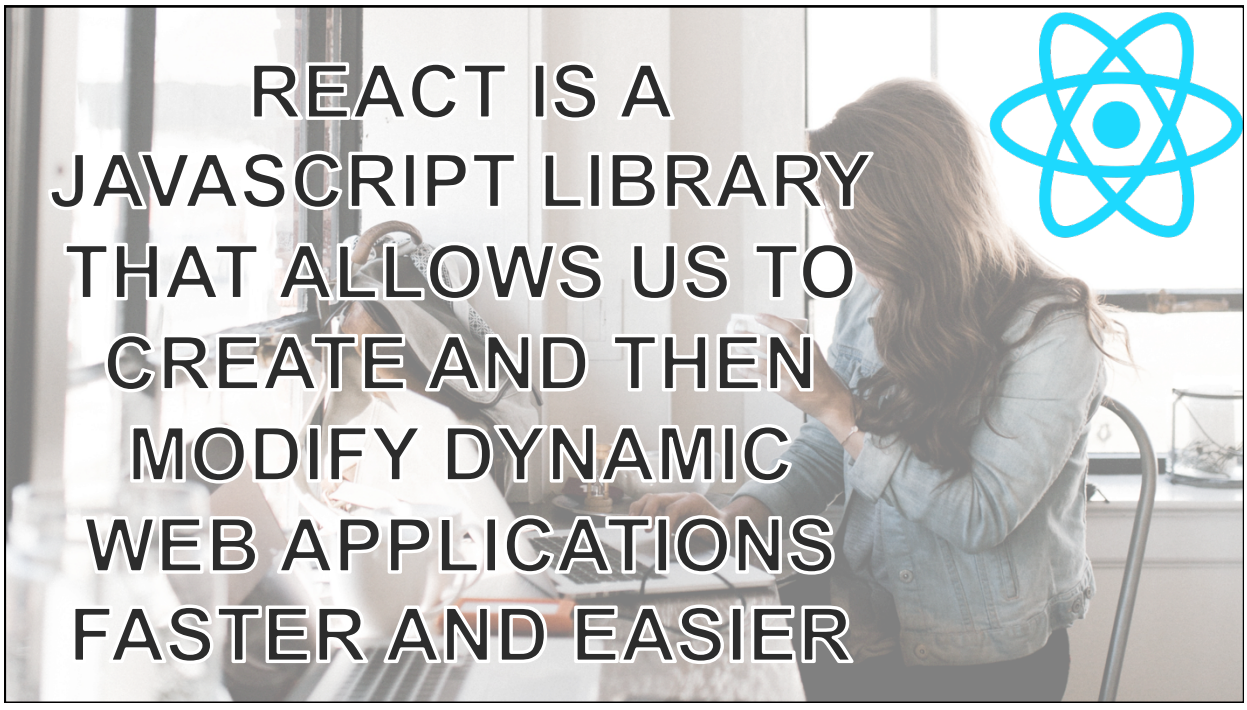
Hello, React

A gentle introduction to React. What is this thing anyway, and why should I learn it?

tl;dr

- What is React? It's a JavaScript library that helps you to create single page web apps faster and more abstractly
- The 3 design philosophies you need to know to understand React
- How React does things so darned fast!
- The secrets of React - What is really happening when an app runs...
- ... And how it was created

What is React?

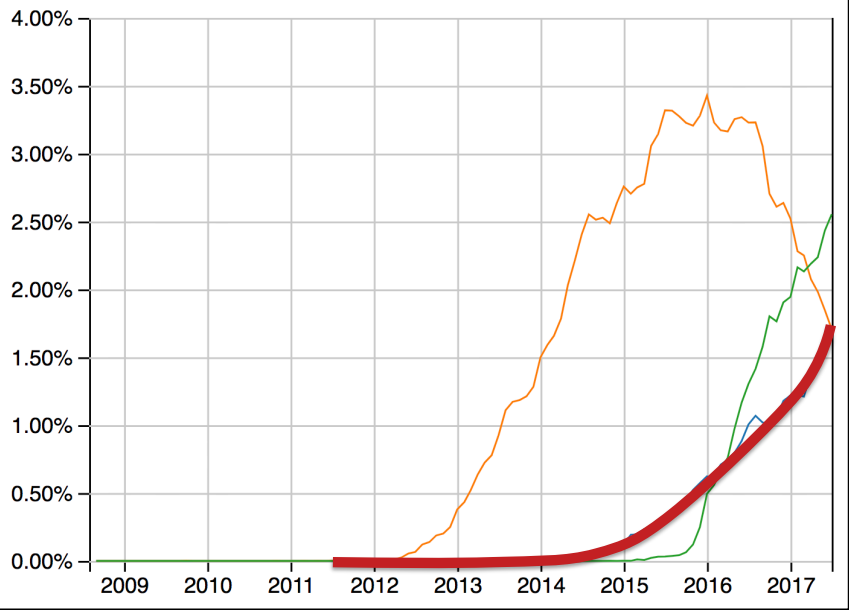


Written in 2011 by Jordan Walke of Facebook's Ads team

facebook

... And soon took over

Released as open source in 2013



The top 3 design principles of React

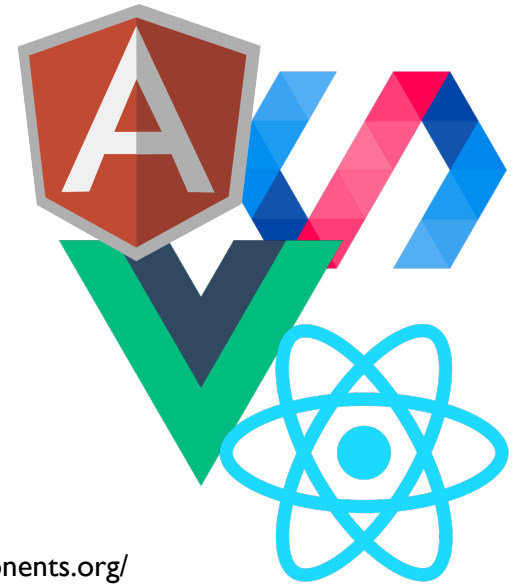
1. Composition of components
2. Events -> state change -> re-render
3. One-way data flow

The web is going to components

- The web component spec is coming*
- We'll all be writing components in a few years

But for now...

- React
- Polymer
- Angular
- Vue.js



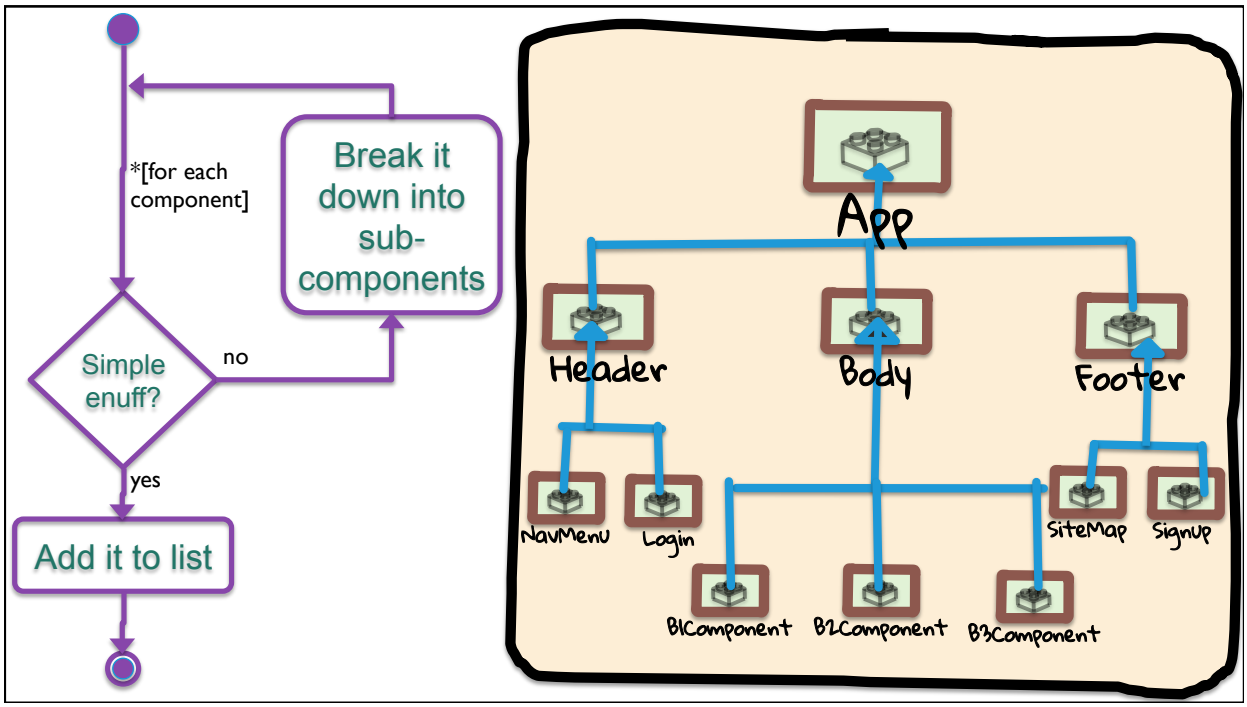
* Find more information on http://www.webcomponents.org/

With React, you'll no longer write pages; you'll write components

Each component is self-contained and encapsulated

- Even styles are local; CSS no longer cascades through components


YOU'LL BREAK DOWN THE PRODUCT INTO SECTIONS. THEN YOU BREAK THOSE INTO SMALLER SECTIONS AND SO ON ...
PHOTO: LEGO


*[for each component]
Break it down into sub-components
Simple enuff?
no
yes
Add it to list
App
Header
Body
Footer
NavMenu
Login
SiteMap
SignUp
B1Component
B2Component
B3Component

In each component you define ...

- How the input parameters (props) should be displayed
- How to react to user interactions (events)

Components should be pure functions when possible.

- React will ... react ... to the data provided.
- The most predicable and debuggable components are those that are pure.

How is React so darned fast?

- React is fast because of three things:
- The virtual DOM
- One-way data flow
- It is lightweight

Reason 1: React mimics the DOM

- JavaScript makes web pages dynamic by altering the DOM.
- But the real DOM is *extremely* heavy because it has to carry so much information in it.
- Because it is so heavy, it is very slow to manipulate.
- Just facts of life in the DOM manipulation world

- So Jordan (et. al.) said "Hey, why don't we create our own little version of the DOM tree,"
- "One that has just enough stuff in it for us to manipulate?"
- "Then we can compare much more quickly and help the browser out with DOM changes!"
- Brilliant idea! FTW!

virtual DOM

- Internal algorithms compare the before and after pictures and determines the smallest amount of changes that can be made. And batches them. And then makes the changes in the DOM.
- Sound like a lot of overhead? Yep.
- But it ends up being so much faster!

- Because it diffs, it only makes changes to the part(s) of the page that have changed.
- Thus it only changes when internal state changes (Spoiler: React tracks 2 kinds of data:
  - Data that is sent into a component and does NOT change
  - Data that can change.
- Hint: Keep your components simple ... don't ever calculate diffs. Just redraw the whole component and let React use the virtual DOM to decide what should redraw and what won't need to.

Reason 2: Data only flows one way

- Data only flows down from parent to child. Never back up.
- Nothing ... not even form fields ... are bound 2-way.
- "So, how do I bind, then?"
- Patience, grasshopper. I promise I'll show you if you come back. It's not easy.

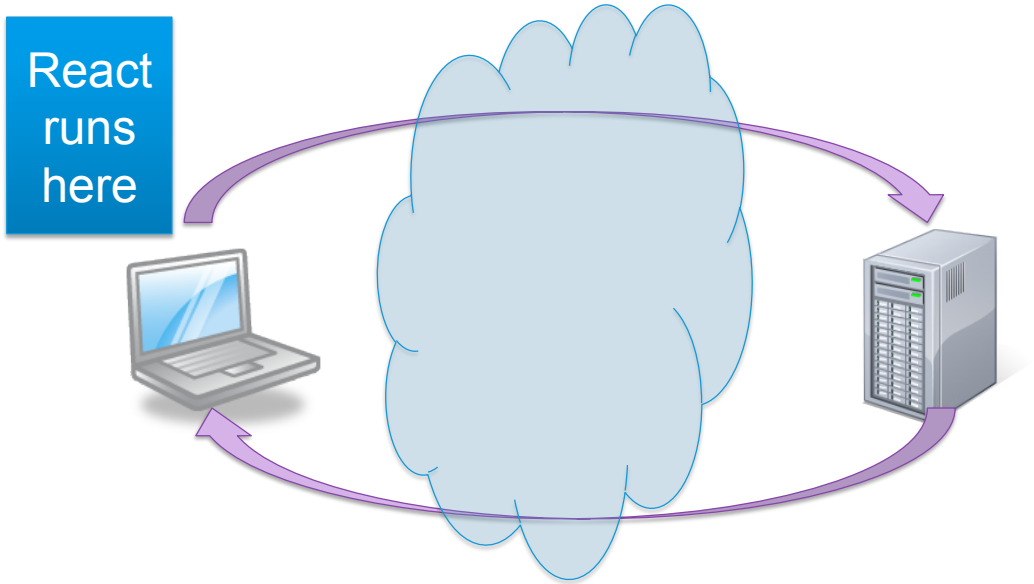
Reason 3: It is lightweight

- It is a library, not a framework
- No built-in Ajax capabilities
- Not opinionated about typing, threading, state management, routing, etc.
- You can add libraries as you like

How a React app works

And how it gets there

The web uses a thin-client architecture



How React works at runtime

1. User requests *any* url from our site.
2. Server responds with index.html
3. client requests bundle.js which is provided and contains all html, css, JavaScript files, libraries
4. Client runs

```
React.render(<App />, someNode)
```

5. This loads your main React component on the page.
6. It loads all subcomponents and so forth and so on.

- Behind the scenes, React uses
- webpack, Babel, npm, ugilfy, and a bunch of other tools
- to transpile, bundle, and minify all your HTML, CSS, and JS into bundle.js.
- All this is automated through scripts and npm.

- Bundle = put all the JS in one big file.
  - Reduces fetches and speeds us up
- Minify = Remove all bytes not needed to run.
  - Reduces size of download and speeds us up.
- Transpile ... Wait, what?

Transpile = converting one dialect of JavaScript into another

- Translating + Compiling = Transpiling
- And why do we need to transpile with React?

Because React uses a dialect called JSX

- Rap, example of JSX here
- No browser in the world understands JSX, so it is transpiled by Babel/webpack/loaders to this:
- Rap, example of React.createElement(...)

React demands that you have a very precise, very complex setup with literally hundreds of interdependent libraries having dozens of version numbers each.

If one small part of it is not configured properly, the system fails.

Wow! all these parts! Wouldn't it be nice to have a way to automate the creation/ scaffolding of the different parts?

Next chapter!

tl;dr

- What is React? It's a JavaScript library that helps you to create single page web apps faster and more abstractly
- The 3 design philosophies you need to know to understand React
- How React does things so darned fast!
- The secrets of React - What is really happening when an app runs...
- ... And how it was created