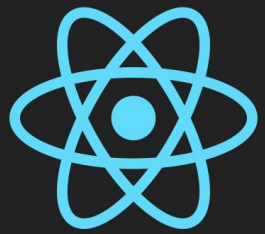
Full Stack IV

Introduction to React



React History

Timeline and usage

React Timeline

- 2011 Created by Facebook
- 2012 Used on Instagram
- 2013 Open sourced
- 2014 Embraced by many large companies
- 2015 React Native released
- 2016 React 15 released

The following large companies now utilize React in production

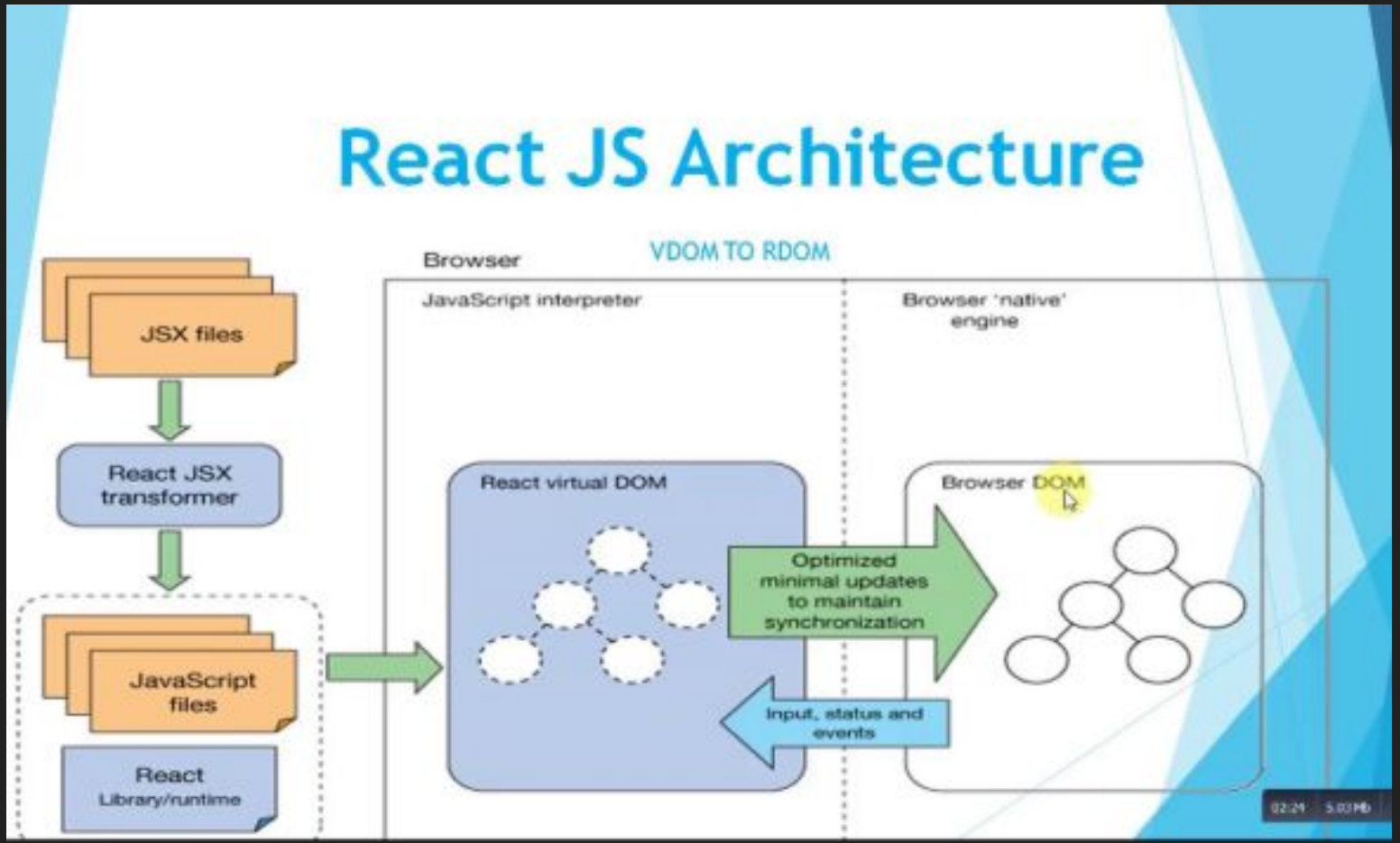
- Facebook and Instagram
- Netflix
- AirBnB
- BBC
- Github
- PayPal
- Reddit
- Yahoo (mail)

Facebook and React

- Facebook used React internally for 2 years before making it open sourced
- Some initially dismissed it because React did not follow popular practices by mixing markup and logic together in a single file
- As time went on, more people embraced React's new component centric approach for separating concerns.
- React is now a mature and stable platform with over 5 years of active development and heavy production usage.
- Facebook is deeply committed to React and employs a full-time React development staff to support the library.

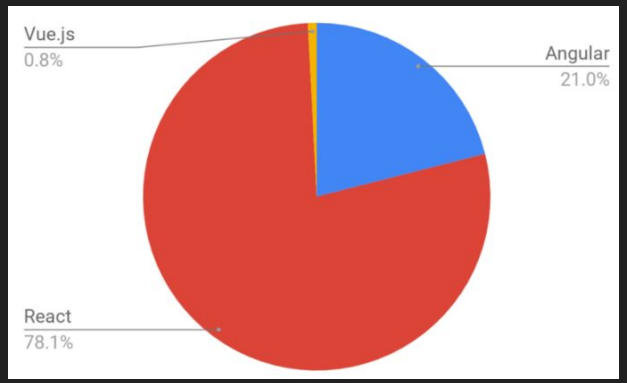
React Architecture

React JS Architecture



React Benefits

Flexibility



Indeed worldwide 2018 trend on 60K job offers

- Learn React once and you can write applications just about everywhere.
- React is remarkably flexible. React embeds fewer opinions than its competition, so it's more flexible than opinionated frameworks like Angular.
- React is a library, not a framework. This library approach allows it to be a remarkably flexible tool.

React Ecosystem

React has grown in popularity and so has its ecosystem. We can use React to build the following:

- To build web applications
- Build static sites using Gatsby and GraphQL
  https://www.gatsbyjs.org/
- Build native mobile applications using React Native
  https://facebook.github.io/react-native/
- Desktop application using Electron
  https://facebook.github.io/react-native/
- Server side rendering using NEXT.js
  https://nextjs.org/learn/
- VR rendering with React 360
  https://facebook.github.io/react-360/

Highly Versatile

- React is highly versatile because the render is separate from React itself.
- For web apps, you call call react-dom to render your components in HTML
- For React Native, you use react-native to render React components into native-friendly code
- For VR apps, you call use react-vr for rendering React components into a virtual reality environment
- More than a dozen other React renderers
  https://github.com/chentsulin/awesome-react-renderer

Easy to Learn API

- React's offers a simple API that's easy to learn. There are few concepts to master and you will rarely need to check the docs.

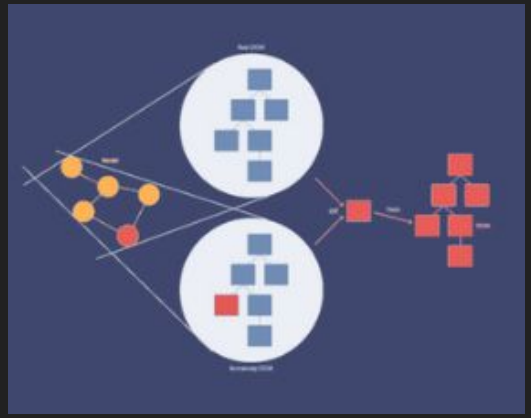
- The concept is basically a function that returns what looks like HTML

```jsx
import React from "react";

const Greeter = (props) => {
  return (
    <div>
      <p>Hello { props.name }</p>
    </div>
  );
};

export default Greeter;
```

```jsx
<h1 color="red">Hello World</h1>

React.createElement("h1", { color: "red" }, "Hello World")
```

JSX compiles to JavaScript

Performance



- JavaScript is fast, but it's DOM (Document Object Model) is slow.
- React team realized updating the DOM is expensive, so to enhance performance they update in the efficient way possible using Virtual DOM.
- When you change data, in behind the scenes React is intelligently figuring out the most efficient way to update the DOM.
- Previously, other libraries would redraw most of the page only when minor changes had occurred.
- React with React DOM weighs only 35K when gzipped and minified. (Preact weighs only 3K)
-

One Way vs Two Way Binding

Magically kept in sync

```
let student = "Ric Flair";

<input
 type="text"
 value={user}
 />
```

Code is more explicit and controlled

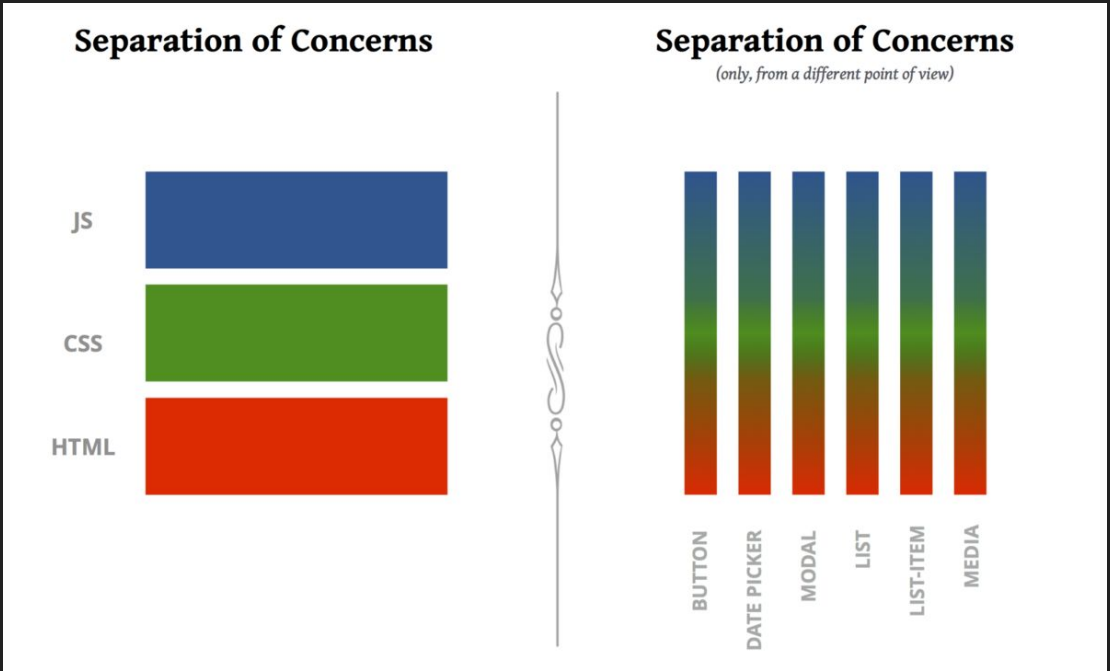
```
state = { student: 'Ric Flair'}

function handleChange(event) {
  this.setState({
    user: event.target.value
  })
}

<input
    type="text"
    value={this.state.user}
    onChange={this.handleChange} />
```

React is JavaScript focused

- React uses JavaScript with HTML, so fewer concepts to learn. Angular and Vue, power up the HTML with their own directive markup

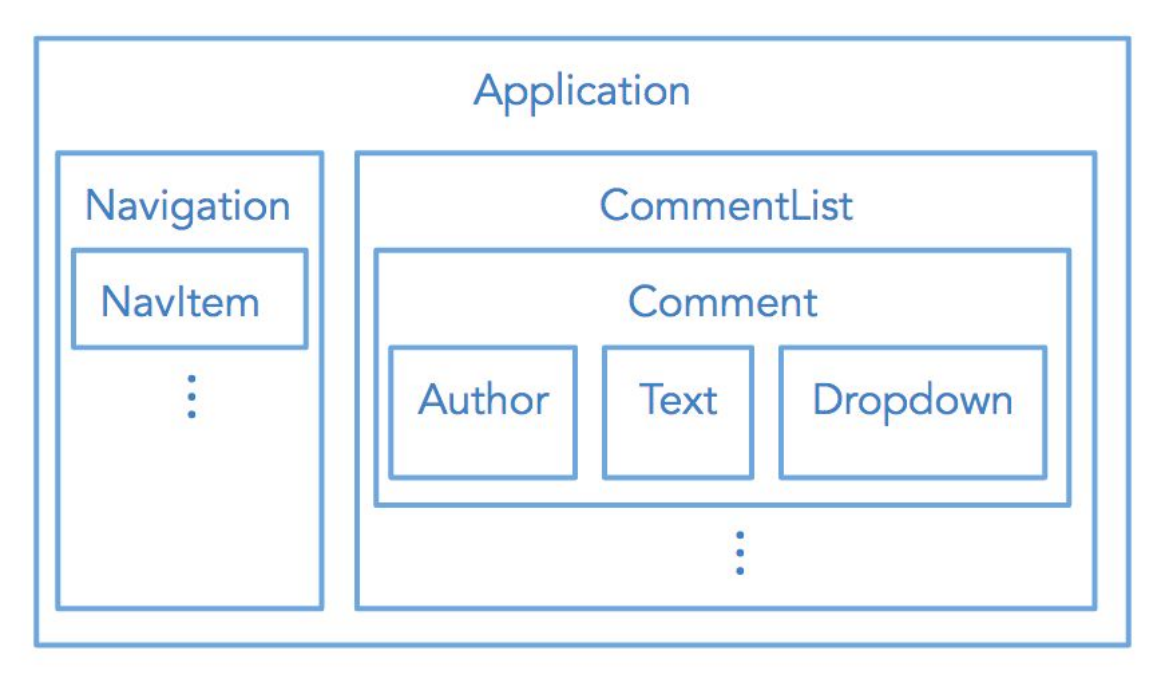
if	ngIf	v-if	if or &&
loop	ngFor	v-for	map
click	(click)	v-on:click	onClick

Separation of Concerns



- Traditionally we would separate the technologies, but React went against this.
- React mixes the JS and JSX in the same file. Each component is a separate concern.

Nested Components



- We can think of a page as a set of nested components. You can have small simple components ie. like button and rating component. Then we can compose the simple components to make a big summary component.
- Migration is easy, because we can replace a small component at a time. Facebook migrated the whole site in this way from PHP to React

Testability

https://jestjs.io/

- Testing on the front end is hard. But React is is very friendly to automated testing.
- You can write small command line unit tests against a single React component in isolation.
- No browser or DOM required in the UI, tests can be written quickly using popular tools like Jest and Enzyme

Library vs Framework

Framework vs Library

- Angular is a framework and contains all building blocks needed for an Angular application.
- React is considered a library, since it's lean and focused on components.

- framework contains more opinions, so you can avoid spending time choosing options
- reduces decision fatigue and there's often less setup overhead
- enforce consistency across applications

- React is significantly smaller than most frameworks, at only around 35K gzipped
- It is small enough to be integrated in existing application so you can slowly migrate
- It doesn't force decisions on you, it allows you to pull in only what you need to keep your app lean and fast

Use only what you need..

React is just a component library and doesn't come bundled with anything, you can pull in whatever else you may need. Angular comes bundled with a full suite of options.

- Components
- Testing
- HTTP Library
- Routing
- Form Validation
- CLI

- Components
- Jest, Mocha
- Fetch, Axios
- React Router
- react-forms
- create-react-app

Decision Fatigue

React is lightweight and unopinionated. They are multiple ways to do the same thing. Many options!

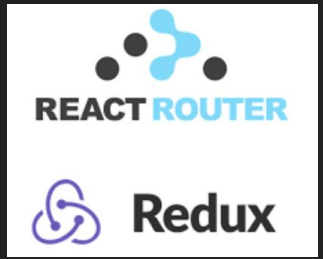
Getting started in React can seem intimidated, because of all the choices. You need to make decisions on the following things:

1. Dev environment

- 100s of boiler plates to configure React apps.
- create-react-app or create-react-native-app *(Auto-testing, Transpiling, Bundling, Linting, Automated build)*

Related Libraries
These are not needed to get started.

- 1. React Router
- 2. Redux



Decision Fatigue cont..

2. Types

- React PropTypes
- Typescript
- Flow.
  (Flow is a static type check for JavaScript created by Facebook)

3. State Management

- Plain React
  (Component state)
- Flux
  (Centralized state ~ Facebook)
- Redux
  (Most Flux popular)
- Mobx
  (Observable state)

4. Styling

- 50 different ways, just use what you know

Features

	React	Angular
Components	✓	✓
Testing	Jest, Mocha	✓
HTTP library	Fetch, Axios	✓
Routing	React Router	✓
I18n	react-intl	✓
Animation	react-motion	✓
Form validation	react-forms	✓
CLI	create-react-app	angular-cli

Hello, React!

All you need for a working React app is a DOM container, an empty tag to mark the spot where you want to display something with React

```
<div id="root"></div>

<script text="text/babel">
  ReactDOM.render(
      <h1>Hello, World!</h1>,
      document.getElementById('root')
  )
</script>
```

- The ReactDOM.render() function renders the <h1> heading element into the div element with id=”root”
- All the heavy work is done by react.js, react-dom.js and babel.js