Before we begin...

- Videos On!

Welcome

Agenda

- Single Page Applications
- Front-End Frameworks
- Thinking in Components
- JSX
- React

Single Page Applications

Let's take a moment...

To have a look at some applications:

- Google Maps
- Instagram
- Twitter
- Telegram / WhatsApp etc.

Their interfaces change, and things happen **without reloading**

These are called Single Page Applications

What are Single Page Applications?

- Apps that don't require reloading
- The page is loaded, then JavaScript takes control
- SPAs typically rely heavily on APIs, AJAX, and DOM Manipulation
  - As well as frameworks to organise the code, because they are very JS-heavy

It's a bit of a misnomer, because there are different "pages" - it's just that the page doesn't refresh. JS just changes it

Pros of Single Page Applications

- They often have a very nice user experience
- They tend to promote the use of APIs
  - Meaning apps are decoupled and back-ends can be used in multiple environments (e.g. web and mobile apps)
- Interactions happen very quickly and updates seem instantaneous
- Loading screens and page transitions are easier
- There is much less load on the server

Cons of Single Page Applications

- They require lots of JavaScript and only work if JavaScript is enabled
- Search Engine Optimization is a little more difficult
- The initial load, if not managed correctly, can take much longer
- They tend to be less secure
- They tend to mean developers have to take control of things they wouldn't otherwise have to (e.g. browser history etc.)
- They are resource-heavy for the browser which can slow performance and put load on devices

Front-End Frameworks

What are Front-End Frameworks?

They are tools that we can download to organise large JavaScript-heavy applications (particularly for Single Page Applications)

They provide structure and common utilities for creating User Interfaces

We follow their patterns and learn their APIs to make our lives easier (this is often described as Convention over Configuration - swim with the current!)

Frameworks for Single Page Applications

- React
- Vue
- Svelte
- Angular
- Ember
- Backbone
- Plus, many more

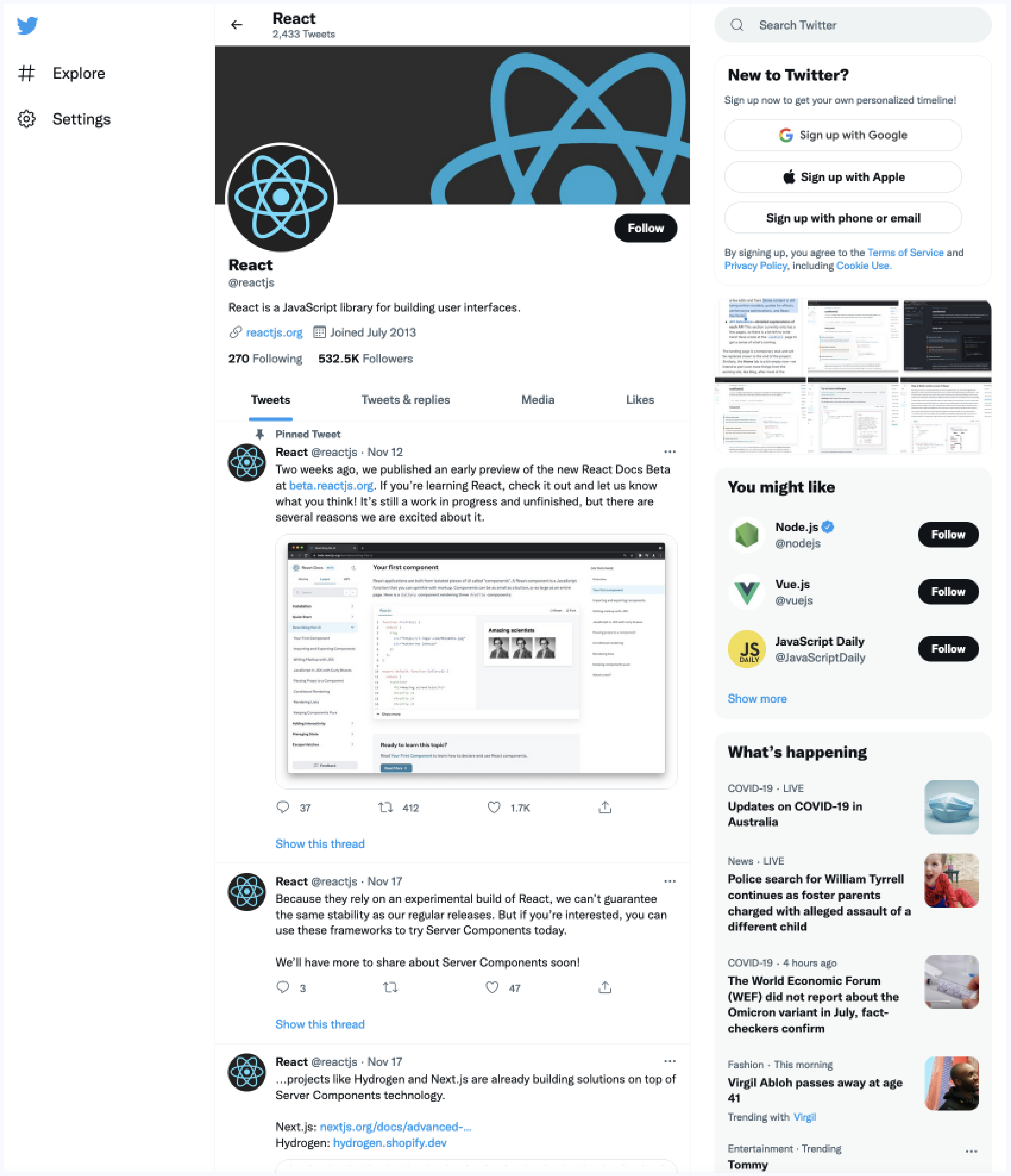
What do Front-End Frameworks provide?

Structure and common utilities to create interactive user interfaces. They can also provide:

- Ways of working with data
- Ways of describing markup (often called Components)

Thinking in Components

Let's take a moment



Explore

Settings

React
2,433 Tweets

Follow

React
@reactjs

React is a JavaScript library for building user interfaces.

reactjs.org Joined July 2013

270 Following 532.5K Followers

Tweets Tweets & replies Media Likes

Pinned Tweet

React @reactjs · Nov 12
Two weeks ago, we published an early preview of the new React Docs Beta at beta.reactjs.org. If you're learning React, check it out and let us know what you think! It's still a work in progress and unfinished, but there are several reasons we are excited about it.

37 412 1.7K

Show this thread

React @reactjs · Nov 17
Because they rely on an experimental build of React, we can't guarantee the same stability as our regular releases. But if you're interested, you can use these frameworks to try Server Components today.

We'll have more to share about Server Components soon!

3 47

Show this thread

React @reactjs · Nov 17
...projects like Hydrogen and Next.js are already building solutions on top of Server Components technology.

Next.js: nextjs.org/docs/advanced-...
Hydrogen: hydrogen.shopify.dev

Search Twitter

New to Twitter?
Sign up now to get your own personalized timeline!

Sign up with Google
Sign up with Apple
Sign up with phone or email

By signing up, you agree to the Terms of Service and Privacy Policy, including Cookie Use.

You might like

Node.js
@nodejs
Follow

Vue.js
@vuejs
Follow

JavaScript Daily
@JavaScriptDaily
Follow

Show more

What's happening

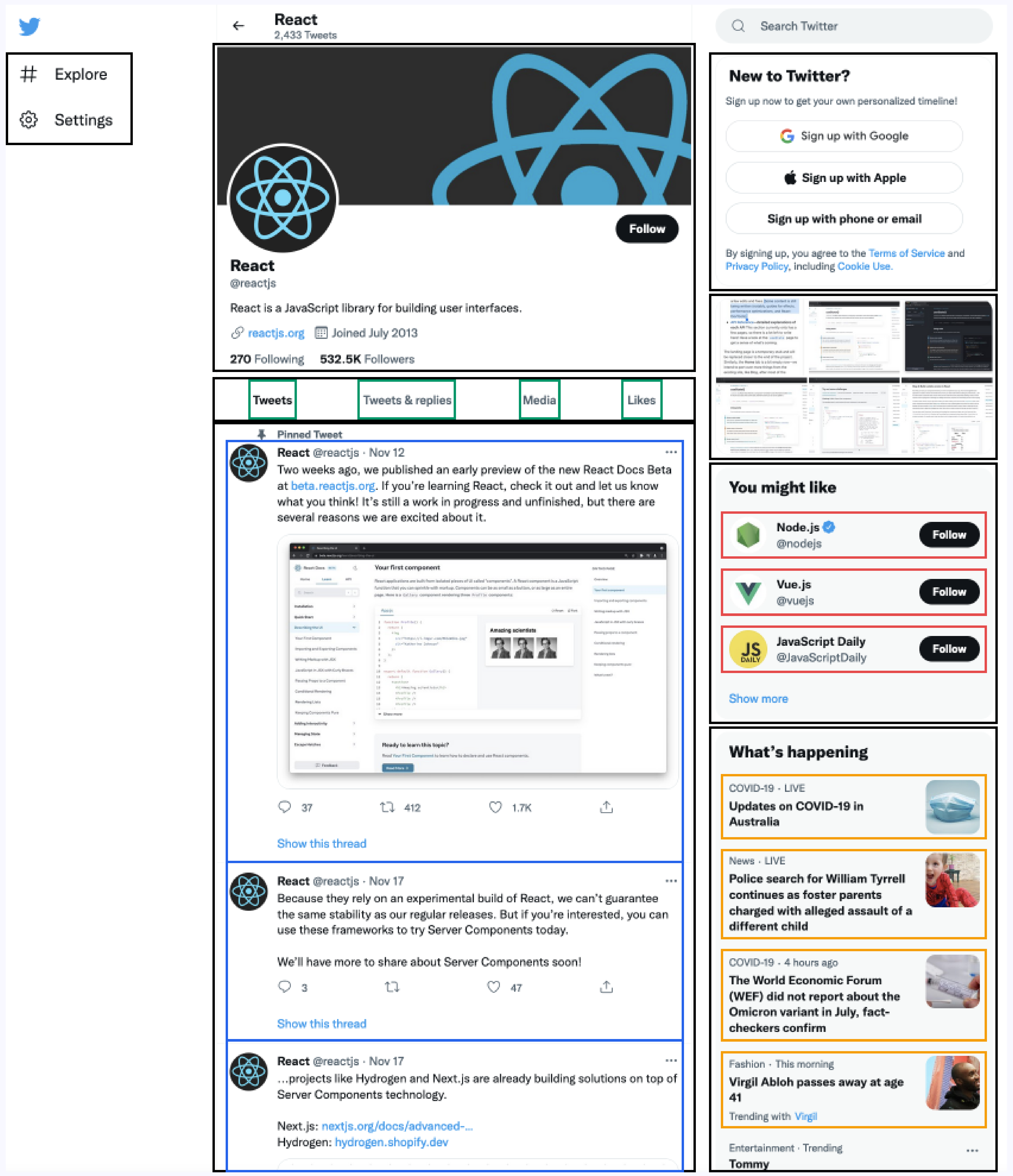
COVID-19 · LIVE
Updates on COVID-19 in Australia

News · LIVE
Police search for William Tyrrell continues as foster parents charged with alleged assault of a different child

COVID-19 · 4 hours ago
The World Economic Forum (WEF) did not report about the Omicron variant in July, fact-checkers confirm

Fashion · This morning
Virgil Abloh passes away at age 41
Trending with Virgil

Entertainment · Trending
Tommy



Explore

Settings

React
2,433 Tweets

Follow

React
@reactjs

React is a JavaScript library for building user interfaces.

reactjs.org Joined July 2013

270 Following **532.5K** Followers

Tweets | Tweets & replies | Media | Likes

Pinned Tweet

React @reactjs · Nov 12
Two weeks ago, we published an early preview of the new React Docs Beta at beta.reactjs.org. If you're learning React, check it out and let us know what you think! It's still a work in progress and unfinished, but there are several reasons we are excited about it.

37 412 1.7K

Show this thread

React @reactjs · Nov 17
Because they rely on an experimental build of React, we can't guarantee the same stability as our regular releases. But if you're interested, you can use these frameworks to try Server Components today.

We'll have more to share about Server Components soon!

3 47

Show this thread

React @reactjs · Nov 17
...projects like Hydrogen and Next.js are already building solutions on top of Server Components technology.

Next.js: nextjs.org/docs/advanced-...
Hydrogen: hydrogen.shopify.dev

Search Twitter

New to Twitter?
Sign up now to get your own personalized timeline!

Sign up with Google

Sign up with Apple

Sign up with phone or email

By signing up, you agree to the Terms of Service and Privacy Policy, including Cookie Use.

You might like

Node.js @nodejs Follow

Vue.js @vuejs Follow

JavaScript Daily @JavaScriptDaily Follow

Show more

What's happening

COVID-19 · LIVE
Updates on COVID-19 in Australia

News · LIVE
Police search for William Tyrrell continues as foster parents charged with alleged assault of a different child

COVID-19 · 4 hours ago
The World Economic Forum (WEF) did not report about the Omicron variant in July, fact-checkers confirm

Fashion · This morning
Virgil Abloh passes away at age 41
Trending with Virgil

Entertainment · Trending
Tommy

What are Components?

Essentially, a big application is a lot of little "components" composed together

Each one of those components are mostly self-contained. They have their own data, and their job is to turn that data into a little bit of markup (e.g. a single Tweet)

Pages broken into components

We want to break big applications into lots of individual components

Ideally, they'll follow FIRST principles, meaning they should be:

- **F**ocussed
- **I**ndependent
- **R**eusable
- **S**mall
- **T**estable

What are Components?

From a technical perspective, components are functions that return markup

They can be written as:

- A function that returns markup (often as **JSX**)
- A class with a render method that returns markup (often as **JSX**)

JSX

What is JSX?

- A syntax extension of JavaScript
  - It's not part of the normal language!
- Something made popular by React
  - It provides us with shortcuts to create elements
- JSX looks like HTML, and we can mostly treat it is as such
  - But it is turned into JavaScript before it reaches the browser
- It's an easy way to describe UI

Why are we talking about it?

Because React uses a lot of things behind the scenes:

- React (mostly) requires a build system and/or transpiler
- React (mostly) requires JSX

What does JSX look like?

```
let element = <h1>Hello, world!</h1>;
```

This data isn't a string, or HTML. It's a syntax extension to JavaScript. Your transpiler/compiler will take that JSX and turn it into regular JS. It'll eventually look something like this:

```
let element = React.createElement("h1", {}, "Hello World");
```

It's just a shortcut

JSX

```
<h1 id="hello">Hello World</h1>;

// Compiles to...

React.createElement("h1", { id: "hello" }, "Hello World");
```

JSX

```
<img src="http://fillmurray.com/400/400" id="bill" />;

// Compiles to...

React.createElement(
  "img",
  { src: "http://fillmurray.com/400/400", id: "bill" },
  null,
);
```

React

What is React?

- An open-source JavaScript library/framework for building applications (particularly Single Page Applications)
  - It was created by Facebook
- It is declarative - we describe patterns and React does the heavy-lifting
- It is component-based - React makes it easy to break applications down into lots of pieces then compose them
- Learn once, write anywhere

What can React do?

Anything! Lots of companies use it - Facebook, Instagram, Uber, AirBnB, Netflix, Pinterest, Shopify, Twitter, Atlassian, Codecademy, Khan Academy as well as many, many more.

It's used in every context, for every type of app

It is certainly the most popular front-end framework - but it does take time to get used to!

What can React build?

One of the big things about React is that you learn it once, and you can write it anywhere:

- Web Applications
- iOS Applications
- Android Applications
- Windows Applications
- Mac Applications

Advantages of React

- Really easy to see the structure of your app
- Very good at managing state
- Performant
- Virtual DOM
- Data Binding
- Easy to test
- Isomorphic (can be rendered server-side)
- Agnostic (you can use it with all sorts of other libraries)
- Learn once, write everywhere

Disadvantages of React

- A big library
- Lots of magic
- It is just the view layer
- Typically requires a transformation step
- A steep learning curve

Installing

Installing React

You'll need a folder with access to NPM (e.g. you have run `npm init`) and it will also need to have Parcel (or something like it)

```
# Install React and React DOM as dependencies
npm install --save react react-dom
```

Our First Component

```
function Hello() {
  return <h1>Hello</h1>;
}
```

Rendering

Rendering

```
// Import necessary dependencies
import React from "react";
import ReactDOM from "react-dom";

// Create a function that returns a single JSX Element
function Hello() {
  return <h1>Hello</h1>;
}

// Render the JSX as HTML in `document.body`
ReactDOM.render(<Hello />, document.body);
```

Interpolation with JSX

```
function Hello() {
  let name = "Jacques Cousteau";
  return (
    <div>
      <h1>Hello {name}</h1>
      <h2>Hello (in capitals) {name.toUpperCase()}</h2>
    </div>
  );
}
```

Curly brackets mean interpolation in JSX (very similar to ${} in template literals)

Props

What are props?

- Props are very similar to parameters in functions
  - They are a way for us to provide data to a component
- They are immutable (meaning they can't change)
- From a parent component, we can pass data down using props
- They look very similar to HTML attributes

Props

```
import React from "react";
import ReactDOM from "react-dom";

function Hello(props) {
  let name = props.name;
  return (
    <div>
      <h1>Hello {name}</h1>
    </div>
  );
}

ReactDOM.render(<Hello name="Jacques" />, document.body);
```

Event Handlers

Event Handlers

```
function MyComponent() {
  function onButtonClick() {
    console.log("The button was clicked");
  }

  return (
    <div>
      <h1>Hello World</h1>
      <button onClick={onButtonClick}>Click Me</button>
    </div>
  );
}
```

State

State

- State is the way we work with data that can change within individual components
  - It is mutable (meaning it can change) data that is local to a component - it can't be accessed by parent components by detail
- It's the way we make our components interactive

Hooks

What are hooks?

- They are the way that we will manage state in our application (that mutable data that is local to each component)
- Hooks are functions that React defines for us (mostly)
- They are run inside a function component
- They maintain their value even when the component re-renders (meaning updates)
- A lot of errors can arise from creating or running hooks within conditionals or loops (so don't do this)

What hooks does React provide?

- useState
- useEffect
- useContext
- useReducer
- useCallback
- useMemo
- useRef
- useImperativeHandle
- useLayoutEffect
- useDebugValue

Destructuring Assignment

What is Destructuring Assignment?

It is a new feature of JavaScript that provides us with a shorthand for accessing data within objects and arrays

For the moment, it needs to be translated into something that is compatible in browsers (we have Parcel doing that for us)

Object Destructuring

```javascript
let person = {
  firstName: "Jacques",
  lastName: "Cousteau",
};

let firstName = person.firstName;
let lastName = person.lastName;

// That's a bit repetitive, isn't it?
// We could replace it with Destructuring Assignment

let { firstName, lastName } = person;
```

Array Destructuring

```
let alphabet = ["A", "B"];

let letterA = alphabet[0];
let letterB = alphabet[1];

// That's a bit repetitive, isn't it?
// We could replace it with Destructuring Assignment

let [letterA, letterB] = alphabet;
```

`useState`

useState

- A function that React provides for us
- We need to import it before we can use it
- It receives an initial value
- It returns an array with two pieces of data:
  - The current value
  - A function to update the current value

useState

```
import React, { useState } from "react";

function ClickCounter() {
  // Set the starting value to be 0
  let [count, setCount] = useState(0);
  function onButtonClick() {
    setCount(count + 1);
  }
  return (
    <div>
      <h1>You have clicked {count} times</h1>
      <button onClick={onButtonClick}>Click Me</button>
    </div>
  );
}
```

Handling User Input

useState

```
function LogInForm() {
  let [email, setEmail] = useState("");
  function updateEmail(event) {
    setEmail(event.target.value);
  }
  return (
    <form>
      <p>Your email is {email}</p>
      <input
        type="text"
        value={email}
        placeholder="Email"
        onChange={updateEmail}
      />
    </form>
  );
}
```

Working with APIs

```
function MovieSearch() {
  const [title, setTitle] = useState("");
  const [data, setData] = useState(null);
  console.log(data);
  function updateTitle(event) {
    setTitle(event.target.value);
  }
  function searchForMovie(e) {
    e.preventDefault();
    fetch(`http://www.omdbapi.com/?apikey=88e15bed&t=${title}`)
      .then(function (r) {
        return r.json();
      })
      .then(function (data) {
        setData(data); // This will re-render
      });
  }
  return (
    <form onSubmit={searchForMovie}>
      <input type="text" value={title} onChange={updateTitle} />
      <button>Search</button>
    </form>
  );
}
```

useEffect

`useEffect`

- This hook allows us to perform "side effects" in our components (meaning things outside of the component itself)
  - Side effects include data fetching, manually changing the DOM and setting up subscriptions
- It is a function that React provides for us, and that we need to import into our projects
  - It receives a callback function and a dependencies array

useEffect

The useEffect callback function is executed based upon on the dependencies array (the second parameter)

- If there is no dependencies array, it will run after the component renders every time
- If there is an empty array as the dependencies array, it will run only once when the component is first rendered
- If the dependencies array contains things, the useEffect callback will run whenever items in that array change

useEffect

```
function MyComponent() {
  useEffect(function () {
    console.log("Runs the first time the component is rendered");
  }, []);
  return (
    <div>
      <h2>useEffect</h2>
    </div>
  );
}
```

useEffect

```
function MyComponent() {
  let [count, setCount] = useState(0);
  useEffect(function () {
    console.log("Runs every time the component is rendered");
  });
  function updateCount() {
    setCount(count + 1);
  }
  return (
    <div>
      <h2 onClick={updateCount}>useEffect: Clicked {count} times</h2>
    </div>
  );
}
```

useEffect

```
function MyComponent() {
  let [time, setTime] = useState(0);
  useEffect(
    function () {
      console.log("Runs every time the `time` is changed");
      setTimeout(function () {
        setTime(time + 1);
      }, 1000);
    },
    [time], // Dependencies Array
  );
  return (
    <div>
      <h2>useEffect: {time}</h2>
    </div>
  );
}
```

React Router

What is React Router?

- It is another NPM package
- We use it control which components are on the page at any given time
  - Essentially allowing us to navigate a single page application is if it were a normal website

Installing

```
npm install --save react-router-dom
```

```
import { BrowserRouter, Switch, Route } from "react-router-dom";

function App() {
  return (
    <BrowserRouter>
      <Switch>
        <Route exact={true} path="/">
          <Home />
        </Route>
        <Route path="/about">
          <About />
        </Route>
        <Route path="/contact">
          <Contact />
        </Route>
      </Switch>
    </BrowserRouter>
  );
}

export default App;
```

```jsx
import { Link } from "react-router-dom";

function Nav() {
  return (
    <nav>
      <ul>
        <li>
          <Link to="/">Home</Link>
        </li>
        <li>
          <Link to="/about">About</Link>
        </li>
        <li>
          <Link to="/users">Users</Link>
        </li>
      </ul>
    </nav>
  );
}

export default Nav;
```

That's all for tonight!

Homework

- Learn about Destructuring Assignment
- Learn about Arrow Functions
- Read about JSX
- Go through this Video Series
- Go through the React Tutorial
  - It will do things slightly differently to how we do it, but it will still cover the concepts
- Read Tinselcity whys:packers

What's Next?

- React
- More React
- More React
- THREE.js

Thank You!