React.js

Office Hours

Install Create React App and Yarn before today's class

```
npm install -g create-react-app
```

https://github.com/facebookincubator/create-react-app

https://yarnpkg.com/lang/en/docs/install/

- In order to avoid continuous NPM/Yarn installs, all of today's activity Solved and Unsolved versions include only a `src` folder. You should scaffold out a React app once with Create React App, and then swap out the `src` folder for each activity. You'll see.

Objectives

- To begin to feel comfortable building static UIs with JSX.
- To gain an initial understanding of the component-based paradigm in ReactJS.
- To dissect and build a few simple examples using ReactJS.
- To deepen understanding of passing props between React components.
- To gain a firm understanding of the concept of child-parent relationships in React
- To be able to programmatically render components from an array of data.
- To introduce the concept of class components and component state.

You can’t learn React in a week

Sorry

You can learn the basics so you can teach yourself the ...rest

You today:

By Friday, you will probably feel like this:

By Monday, though:


MAKE GIFS AT GIFSOUP.COM

Enough already!

What the hell is React?

React is an open-source Javascript Library developed by Facebook specifically for the task of developing User Interfaces.

React was developed for the purpose of building large apps with data that changes rapidly

React relies on a component-based architecture.
Each element of the UI is treated as sub-components of larger components

You’re like,

BFD

I’ve got JQuery.

Open Facebook.

Yes, your teacher is telling you to go to Facebook.

Can your JQuery do this?

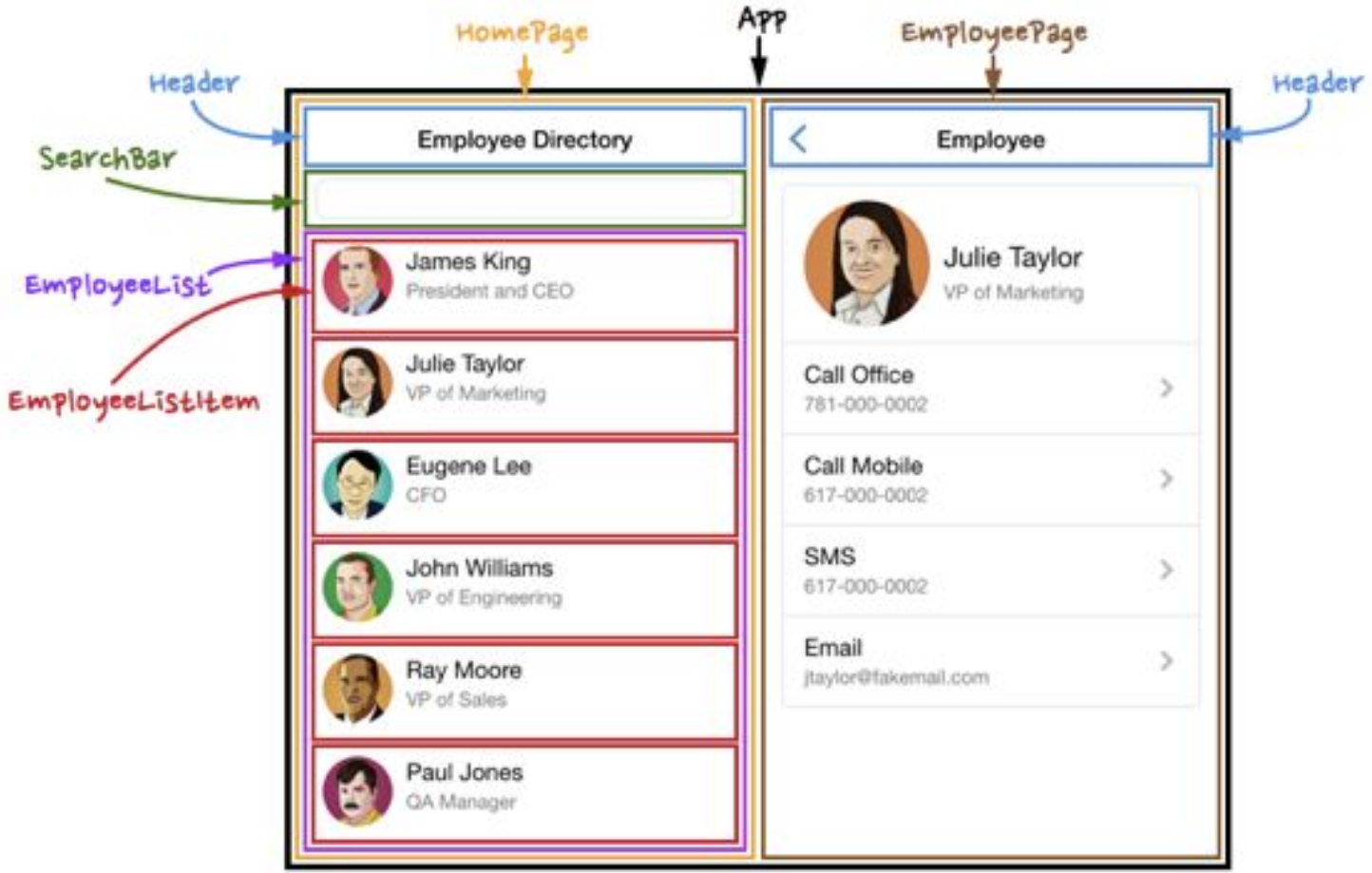

Search Facebook
Ahmed
Home
Ahmed Haque
Edit Profile
FAVORITES
News Feed
Ads Manager
Messages
Events
Photos
Rice Coalition on Hu...
RCoHH Owls Help C...
Rice Muslim Student...
Grassroot Islam - H...
Saved
Sale Groups
PAGES
Sacred Geometry R...
Pages Feed
Like Pages
Create Ad
Create Page
GROUPS
A&M Consolidated H...
Houston Startups
Discover Groups
Create Group
FRIENDS
Close Friends
APPS
Live Video
Update Status
Add Photos/Videos
Create Photo Album
What's on your mind?
Friends
Post
Mother Jones
21 mins
Newsflash, Republicans.
I didn't vote for this Obamanation
A new poll has really bad news for Republicans
MOTHERJONES.COM
274
Like
Comment
Share
Ahmad Razi ▸ Rice Muslim Student Association (Rice MSA)
37 mins
YOUR PAGES
Rice Coalition on Hunger a...
How to Respond to Messages
Help customers to respond to their requests for Rice Coalition on Hunger and Homelessness's services, products or offers
Learn More
Latest Unread Message
Olga Carolina Avila Montes
This Week
2
Post Reach
7
Recent Posts
See More
4 events this week
Osman Athar's birthday is
TRENDING
Stovall Middle School
Nick Foles
Eric Williamson
Thanks!
Hey Ahmed! Just trying to reconnect to see how things are progressing in the NYI Give me a call when you get a chance: 713.306.9799
Tell Sarna hi for me!
Sent from Messenger
Ripley Neighborhood Center
Ali Bin Ashan replied to Ricardo N Santos's comment.
Abdul-Rahman Raji likes Jed Pixley's post.
Hassan Aljazi likes ISGH River Oaks Islamic Center's link.
Sarah Chaudhry Crosby replied to Jaya McSharma's comment.
Ornsiree Junchaya likes Macha Cauchois's photo.
Shiyam Galyon likes Dr Joanne Liu.
Lisha Huang commented on Narcity Vancouver's link.
Mac Kelly reacted to Sharon Logan Kelly's post.
Maggie Wong
Maria Granados
Michelle Flan
Mick Walter
Nelson Vanegas
Noha Mohamed Helal
Search

Imagine building Facebook with JQuery.

Components

Using React, UI elements are broken down into re-usable sub-components.

Each sub-component behaves in a way that is fully contained


HomePage
App
EmployeePage
Header
Header
SearchBar
EmployeeList
EmployeeListItem
Employee Directory
Employee
James King
President and CEO
Julie Taylor
VP of Marketing
Eugene Lee
CFO
John Williams
VP of Engineering
Ray Moore
VP of Sales
Paul Jones
QA Manager
Julie Taylor
VP of Marketing
Call Office
781-000-0002
Call Mobile
617-000-0002
SMS
617-000-0002
Email
jtaylor@fakemail.com

Components

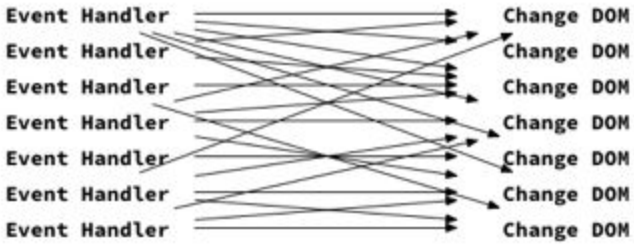
- Layout and Logic are kept bundled together in a self-contained package.
- Components can easily be re-used in various points in the application without needing to be re-coded.
- Components can be more easily tested. (i.e. having one re-usable component means only one UI element needs to be tested).
- For complex applications each of these can be critical in finding bugs and saving time.

Why Separate UI Elements Into Components?

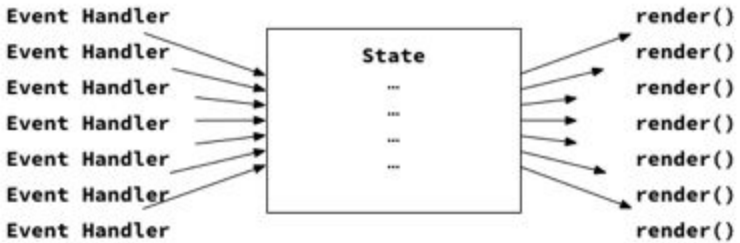
- Components allow us to logically decompose a UI into different parts
- They allow us to easily re-use these parts without re-coding
- Component-based applications are easier to test
- All of this can help us find bugs and save time

Why can’t I just JQuery everything?

jQuery Style



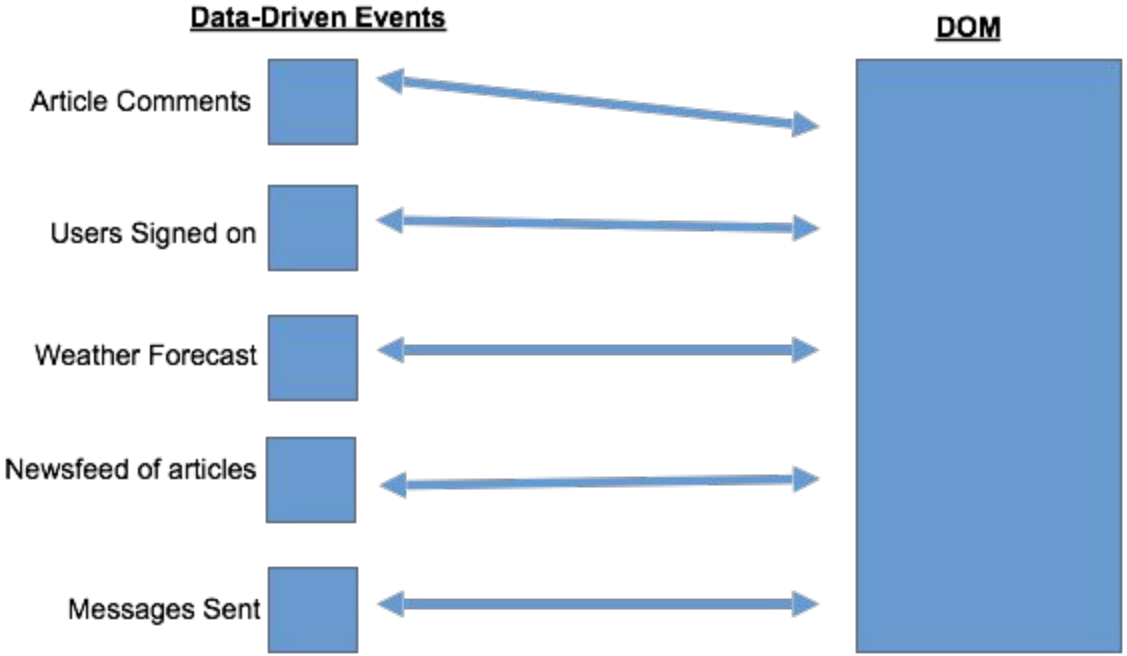
React.js Style



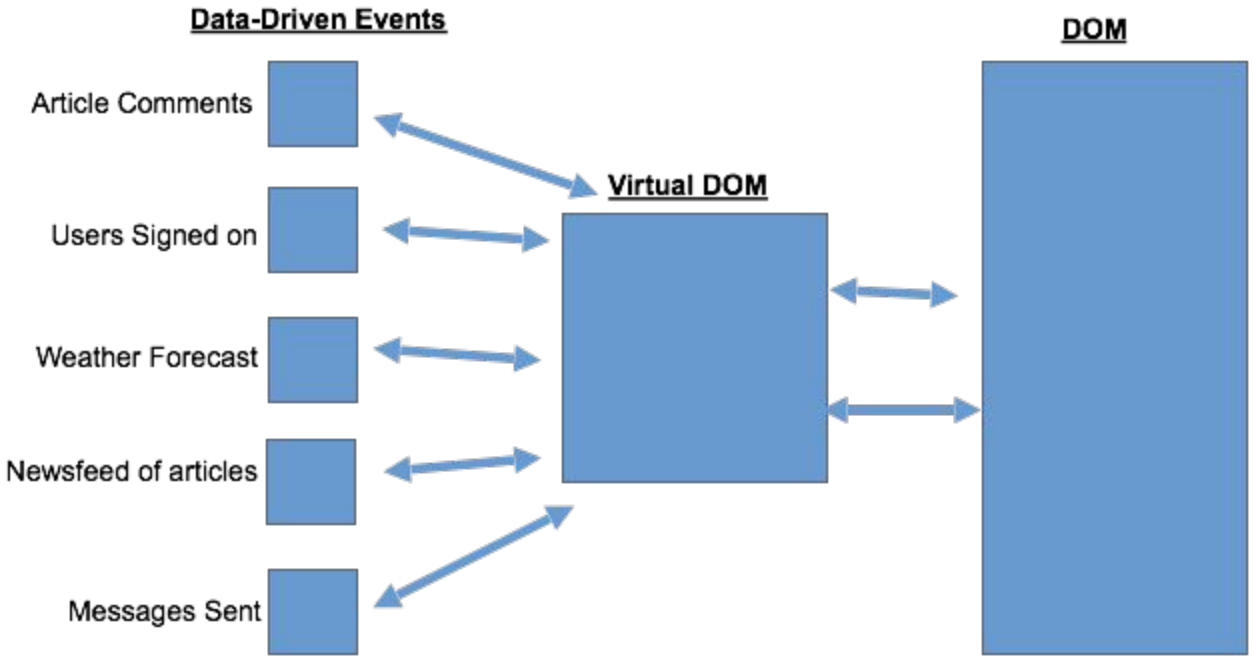
Why Can’t I Just JQuery Everything?

- Jquery
  - Without an organizational structure, code quickly become a series of erratic DOM Manipulations
- React
  - In contrast, React utilizes a state or Virtual DOM that acts as a middle-person with pre-defined rules for how each component will behave.

Data Shifting Applications


Data-Driven Events
DOM
Article Comments
Users Signed on
Weather Forecast
Newsfeed of articles
Messages Sent

Constantly manipulating the DOM is S.L.O.W.


Data-Driven Events
DOM
Article Comments
Virtual DOM
Users Signed on
Weather Forecast
Newsfeed of articles
Messages Sent

Virtual DOM serves as intermediary


Dom

React: Pros && Cons

- A component-based architecture encourages us to break our application's UI into reusable components, allowing us to build more quickly through code reuse
- Our application's UI reactively updates in response to changes to its state, reducing the amount of DOM manipulation code we need to write ourselves
- Can be be used to build applications on the web, on the server, and in native applications
- Arguably easier to learn and more popular than other competing front-end JavaScript libraries and frameworks

- React is just a view library concerned with rendering user interfaces. We'll have to pull in other libraries to accomplish things such as HTTP requests
- Setting up React can require more configuration than other libraries

Tools

webpack

webpack is a module bundler for modern JavaScript applications. When webpack processes your application, it recursively builds a dependency graph that includes every module your application needs, then packages all of those modules into a small number of bundles - often only one - to be loaded by the browser.

https://webpack.js.org/

webpack allows us to modularize our front end code in the same way we do in Node with modules (require, module.exports). webpack also allows us to apply various transformations on our assets through the use of plugins, such as…

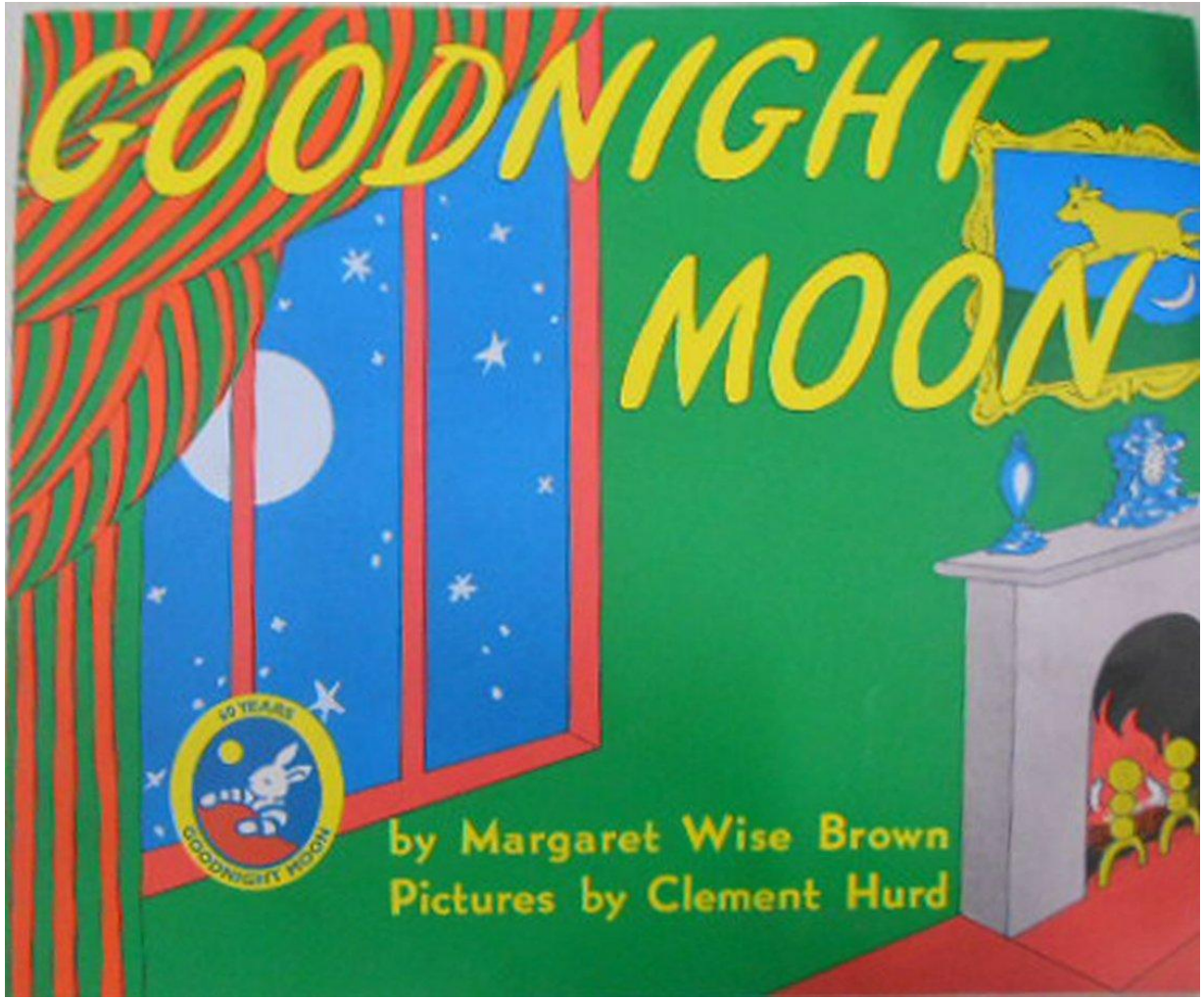
Babel

Babel is a JavaScript compiler. It allows us to transform next generation JavaScript code (ES6, ES7, ES8) into ES5 JavaScript code that most browsers will understand.

Additionally, developers can extend the JavaScript language syntax and create their own features, or even entire languages that compile down to regular JavaScript by creating Babel plugins.

React makes use of both next generation JavaScript, and JavaScript syntax extensions.

Now it’s time to RTFM


GOODNIGHT MOON
50 YEARS
GOODNIGHT MOON
by Margaret Wise Brown
Pictures by Clement Hurd

Activity: RTFM (20 min)

See 01-Stu-DocumentationQuestions/README.md for instructions

What are Create React App and Yarn?

Create React App is a new officially supported way to create single-page React applications. It offers a modern build setup with no configuration.

Yarn is an open source JavaScript package manager built by developers at Facebook, Exponent, Google, and Tilde. It is meant to be a fast, secure, alternative to the Node Package Manager (npm) that can install previously downloaded node modules with limited or no internet connection.

What are the benefits of using React?

Fast Learning Curve

React is more complicated to learn and use than jQuery, but this is an unfair comparison because the two aren't even in the same league. Compared to other competing JavaScript libraries and frameworks such as Angular or Ember, React has much less code that we as developers need to memorize or worry about, making it much faster to learn.

React is Component Based

With React we can build encapsulated components that manage their own state, then compose them to make complex UIs. Since our components are written in JavaScript instead of HTML, we can easily pass data into our application and keep our state out of the DOM.

"Learn Once, Write Anywhere"

React only tries to be the view layer of our application (the V in MVC). Because of this, it makes no assumptions about the rest of our application, and can be rendered on the client, on the server using Node, even in native mobile applications with React Native.

What is a React component?

Components let you split the UI into independent, reusable pieces and think about each piece in isolation. Components can be defined as JavaScript functions or ES6 classes. Essentially components are responsible for rendering some part of an application's UI.

What is the significance of the ReactDOM.render() function?

`ReactDOM` is a package separate from the `React` package that provides APIs for DOM specific methods. `ReactDOM.render` takes a React component, or tree of React components and (eventually) renders them to the DOM. For the most part, we generally only run this method once per React app to render a single root component containing all of our other components.

What is JSX? Why does Facebook recommend it?

JavaScript XML

<Thing />

JSX is a syntax extension to JavaScript that *looks like* HTML.

Facebook recommends JSX because it's familiar HTML-like syntax makes it easy to describe the UI a component should produce. It is possible to write React applications without using JSX, but the process is less intuitive.

What does Facebook recommend as the “best way to start building a new React single page application”?

They recommend everyone begin with Create React App. Normally, setting up a robust React application from scratch requires a deep understanding of build tools such as Webpack (used for bundling an application's assets) and Babel (used for transpiling JSX and ES6+ code to plain, widely supported, ES5 JavaScript).

Thankfully, we can use Create React App to quickly scaffold a React app in seconds with all of the features and configuration we'd need for *most* applications.

What is Babel? And what role does it play in converting JSX into vanilla JavaScript?

Babel is a JavaScript compiler. It takes next generation JavaScript (ES6/ES7/ES8) and compiles it down to widely supported ES5 JavaScript code. It also compiles JSX code down to regular JavaScript functions that browsers can understand. Developers can also extend the JavaScript syntax and add on their own features, and share them with others in the form of a Babel plugin.

What is the significance of { } curly braces in JSX?

The { } single curly braces inside of JSX work similarly to the {{ }} double curly braces in Handlebars:
they allow us to pass values and expressions into our view.

What is a component prop?

A prop is essentially a function argument that's passed into our component from outside and can be used inside of it. For example, we could write a component for a button that renders different types of buttons depending on the prop it's passed.

Takeaways

- We use components as a way to separate our application into reusable pieces of its UI. This allows us to think about each part of our application's interface in isolation and allows us to quickly build new views into our apps over time since we'll be able to reuse component's we've already created.
- Using JSX is optional but most React projects make use of it. JSX allows us to describe the UI our components using familiar HTML-like markup in our JavaScript. This allows us to quickly describe and understand the UI our components will produce as well take advantage of the full power of JavaScript, rather than be limited to the features available in a templating language such as Handlebars.

Demo: Create React App

Follow along…

- Navigate to your projects directory. For me, it's apps
- $ create-react-app reactpractice
  - To scaffold out a React app with create-react-app, we run `create-react-app` followed by a name for our application
- cd reactpractice
- yarn start
- http://localhost:3000/

/src/

- Create React App generates a `src` folder. Anything inside this folder gets processed by Babel, and then bundled into `bundle.js` by webpack.
- This is where we store our React components.
- App.js
  - This file is producing JSX, which corresponds to the web page we rendered in the browser.
- /index.js
  - This is where execution of our React application begins. In this way, it is similar to the `server.js` file we've been starting all of our node applications with. In order to run any code in our React app, it needs to be directly written here, or else required/imported here.
- registerServiceWorker.js
  - this is a fairly new addition to Create React App. We don't need to understand the contents of this file, but it helps cache API responses to help make our app perform better for users with poor internet connections.

/public/

This is primarily for containing the `index.html` that will eventually be served to users visiting our application.

Note:

- This will contain our entire rendered, React application.
  - When we build our React application for production, a `bundle.js` file containing all of our application's JavaScript is generated and added inside of the `index.html` file. We can also store files in this folder which we don't want to be processed by Webpack and Babel. For example, we can place static CSS files here or link to external CDNs in the `head` tag of the `index.html`.

Takeaways

- We're going to be writing most of our code inside of the `src` folder.
- The "entry" file to our React application will be the `index.js` file.
- We start our React app in development mode with the command `yarn start`. This means our app will live update as we change it, which is why we're running our app on a server.

Activity: Hello React (10 min)

With a partner, see 02-Stu_HelloReact for instructions

src/components/HelloReact.js

What's happening inside of `src/components/HelloReact.js`? How does it relate to the content being rendered to the browser?

The `HelloReact` function returns some JSX describing the UI we eventually want this particular component to be able to render to the document. If we were to change the JSX being returned by this function, and we were running our React app in dev mode (when we run `yarn start`), the document would auto update inside of our web browser without us having to refresh. Create React App uses a Webpack development server that auto updates the view as the content changes.

The `HelloReact` function is an arrow function. Can you modify this function so it uses a regular function instead (one that uses the `function` keyword)?

```
function HelloReact() {
    return (
    <p>Hello World!</p>
    );
}
```

Under the hood, Create React App utilizes Webpack, which passes our code through Babel, which takes the ES6 code (arrow function) and the JSX code we currently have written and translates it to regular widely supported ES5 JavaScript code.

src/App.js

In this file we define another component named `App`. It's common to have multiple components that fit together inside of a React application. We'll typically compose all of the top level components inside of our `App` component.

What does the `App` function return?

For now, the `App` function or component just returns our `HelloReact` component.

Why do we import the React library? We aren't using the React keyword anywhere. Is it possible to remove this and still have our code work?

Whenever we use JSX inside of our JavaScript, we need to import the React library. When Babel translates our `App` component's JSX code to plain old JavaScript, it looks like this:

```
var App = function App() {
  return React.createElement(HelloReact, null);
};
```

See how the `React.createElement` method is used? Because the plain JavaScript code uses this method, we get an error when compiling if we don't import React.

src/index.js

Do you remember what the purpose of `ReactDOM.render` is? What is it doing?

We use `ReactDOM.render` to render a single component or tree of components to the DOM. In our case, `App` is the root of our component tree (it renders all of our other components inside).

Instead of splitting our files up into `App`, `components/HelloReact` and `index`, is it possible to just write our entire Hello World app in the `index.js` file?

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

const HelloReact = () => (
  <p>Hello World!</p>
);

ReactDOM.render(<HelloReact />,
   document.getElementById("root"));
```

* React is fairly unopinionated, so we can accomplish the same thing in a variety of ways. There are more conventions and best practices than there are right and wrong ways to do things.

Activity: HelloDiv (10 min)

See 03-Stu_HelloDiv for instructions

Somebody Slack their solution to #class-instruction

Review: HelloDiv

HelloDiv is exported and rendered inside of `App`.

App is exported and then rendered inside of `index.js` as the first argument to the ReactDOM.render method.

The second argument to the ReactDOM.render method is the real DOM element that our React application should be rendered inside of.

HelloDiv is a JavaScript function; it returns some JSX

- A JavaScript function can only return one value. To have separate groups of JSX being returned outside of a parent element would be like trying to write a JavaScript function that returns multiple different variables at once.

Refresher

- In React, we structure our code into components.
- A component is a JavaScript function that describes some part of our application's UI.
- Inside of our components, we describe our application's UI using JSX: a markup syntax that resembles HTML.

Wait for it...

Demo: Hello Bootstrap

See 04-Ins_HelloBootstrap

$ yarn start

http://localhost:3000

Style, yo?

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css"/>

public/index.html

Let’s look under the hood:
src/components/HelloBootstrap.js

What is the key difference between this JSX and standard HTML?

className

`className` is used instead of `class` because `class` is a reserved word in JavaScript, similar to `var` or `for` or function or `doge`.

Kidding. There’s no doge in JavaScript.


made with dwigif.appspot.com

JSX isn't HTML, so there are a few caveats. JSX is sugar syntax for calls to the `React.createElement` method, and according to Facebook's React Docs:

"JSX is closer to JavaScript than HTML".

RTFM https://facebook.github.io/react/docs/introducing-jsx.html#specifying-children-with-jsx

Activity! Hello Bootstrap (10 min)

See 05-Stu_HelloBootstrap for instructions

Review: Hello Bootstrap

- If we want to render multiple JSX elements, they needs to be contained within a single parent element, such as a div
- Void elements, such as input tags, are represented by JSX tags with a self-closing forward slash, i.e. <input />
- We need to import the react library anywhere that we are utilizing JSX.
- We use className instead of class because class is a reserved word in JavaScript.

Review: Hello Bootstrap ### Bonus

- The main takeaway here is that we can define our components in different files and compose multiple components inside of another component (Navbar, Jumbotron, and Panel are rendered inside of App).
    - This is powerful! Instead of having to rewrite every UI element from scratch using individual HTML tags, we could write a component once and reuse it across our entire application.

Demo: JSX Variables

See 06-Ins_JSXVariables

src/components/JSXVariables.js

- We can write comments inside of our JSX using JavaScript's multi-line comment syntax inside of curly braces. If we were to try and write regular JavaScript comments inside of our JSX, the code would break.
- In addition to being able to write comments inside of the curly braces, we can also embed JavaScript expressions. This works similarly to the {{ }} double curly braces we've worked with in Handlebars.

Activity! JSX Variables (10 min)

See 07-Stu_JSXVariable for instructions

Review: JSX Variables

Demo: CSS

See 08-Ins_CSSDemo

http://localhost:3000

- Header.js
  - Whenever we import a stylesheet, an internal style tag is created and added to the document.
  - Note how Header is using the classes defined in the Header.css stylesheet it's importing.
- Panel.js
  - We are also able to add inline styles to JSX elements.
  - Note
    - React expects the style property to be an object, rather than a string.
    - Numerical values will default to pixels if not declared as strings
    - Properties are camelCased

Advantages?

Using a style property, our styles are scoped to the components they're attached to, meaning we won't have any issues with CSS scoping or naming collisions.

Because our styles are defined using JavaScript, we can can write JavaScript logic to determine what our styles should be at any given point. This will be useful as we begin to build more dynamic apps.

We could define our component's styles inside of a CSS stylesheet, we could use inline styles, or we could use some mixture of the two. It's up to us to decide how we want to organize our component styles, but there is value in being consistent in whichever approach or hybrid approach we take.

Activity! CSS Props

See 09-Stu_StyleProp for instructions

Lunch O’Clock

React Recap

What are the building blocks of a React application?

Components

Components allow us to split the UI of our application into independent reusable pieces, and think about each piece in isolation.

The Power of Components!

By separating elements out into components...

- Layout and logic are kept bundled together in a self-contained package.
- Components can easily be re-used in various points in the application without needing to be re-coded.
- Components can be more easily tested. (e.g. having one re-usable component means only one UI element needs to be tested).

What is the name of the HTML-like markup syntax we use to describe the UI our components should produce?

JSX

JSX

- JSX is a preprocessor step that adds XML syntax to JavaScript. You can definitely use React without JSX but JSX makes React a lot more elegant.
- While JSX may look like HTML, there are a few gotchas to be aware of. React’s documentation does a great job of listing these.

How can we embed JavaScript expressions/values inside of JSX?

{ }

JSX curly braces {} are used to embed JavaScript expressions.

What library and method do we use to render our application’s root component to the DOM?

ReactDOM.render

ReactDOM is a library separate from React with methods for working with the DOM.

We can conceptualize React components as JavaScript functions.

It's a component's job to describe and return some part of our application's UI.

If a component is a function that returns some data, what else might a component be able to do?

Receive arguments

This allows us to write components that behave differently based on the arguments that they receive.

We call the arguments we pass into React components `props`

Demo: Props

See 10-PropsDemo

This example is rendering a simple Bootstrap alert element.

src/components/Alert.js

- Every component has access to a props argument. Props is always an object containing all of the values passed to the component.
- We're using props.type to determine what the evaluated className of the div element is.
- This component renders props.children between it's div tags.
- since props.type is equal to "danger", then the computed className of the div in the Alert component is "alert alert-danger". This Bootstrap class is providing our component its styles.
- The Alert component is also receiving a children prop with a value set to "Invalid id or password" — the same message being displayed inside of the rendered Bootstrap alert element.

src/App.js

We have 2 ways of passing props into a component:

1. We can set an attribute to the rendered component's tag.
    a. We're passing a type prop equal to danger.
2. We can give a component a sibling tag and pass an expression between the tags.
    a. We don't name this prop, it is automatically set a children key.
    b. We're passing a children prop equal to "Invalid user id or password".

We can change what is rendered by the Alert component by passing it a different type and children prop

Props are the primary means by which we pass data around our React apps.

props allow us to customize our components so that we can reuse them in different situations.

React utilizes a unidirectional data flow, meaning data only flows one direction: from the top down, parent to child.

If a prop inside of our component isn't what we expect it to be, where could we look to find out why?

At the component's parent

In this example, `App` and `Alert` have a parent/child relationship. `Alert` is being rendered inside of `App` and `App` is passing props to `Alert`.

Activity! Calculator Props

In this activity students will work with a partner to write a component that accepts props, performs arithmetic and renders the result.

Review: Calculator Props

Activity: Props Review

See 11-Stu_PropsReview

Review: Props Review

We created a custom component, FriendCard. Note the file structure: our component contains an index.js, a style sheet and the component itself.

- src/components/FriendCard/FriendCard.js
  - we use the props argument to access all of the values passed into the FriendCard component.
- src/App.js
  - We import friends.json
  - In a real application, where might all of the friend JSON data come from?
    - Normally we might receive the friend JSON from an AJAX request, and probably won't know ahead of time which friends will need to be rendered.
- src/components/FriendCard/index.js...

What is this line of code doing in our component?

```
export { default } from "./FriendCard";
```

The index.js file inside of each component's folder allows us to export the component from the index.js, in addition to its own file

Whenever we require or import a folder instead of a file, the folder's index.js file is required or imported by default

This allows us to keep our paths for importing these components short.
For example, we can do…

```
import FriendCard from "./components/FriendCard"
```

...instead of...

```
import FriendCard from "./components/FriendCard/FriendCard"
```

Giving all of our components their own folder is another option for organizing our React apps. Each folder could contain any CSS or other dependencies the component will need.

Activity! Component Map (10 min)

See 12-Stu_ComponentMap for instructions

Review: Component Map, Basic

- src/App.js
  - We create an array, `groceries`, of objects
  - `groceries` is passed to our List component making it available there as `props.groceries`
- src/components/List.js
  - Leveraging props, we declare a List function and pass it `groceries` array as a JSX {variable} inside a `ul` element
  - We then `.map` each item in the array to an `li` element by `id` using the React `key` attribute and insert each items name between the `li` tags

Key, Pun Intended, Takeaway

We can use the map method to loop over an array and return a new array of elements inside of JSX curly braces. React will then render each element in the resulting array.

RTFM https://facebook.github.io/react/docs/lists-and-keys.html

Review: Component Map, Bonus

We create a notPurchased array by filtering props.groceries for groceries which have a purchased property set to false.

While the map method returns a new array the same length as the original, the filter method returns a new array containing only the elements whose callback functions return truthy values.

We still need to use the map method to actually render the li elements. But we first filter for groceries which haven't been purchased, and then map over the new array, rather than props.groceries.

Stateless vs. stateful

Stateless

- What we've been working with so far are known as stateless, functional components. Sometimes called "dumb components".
- These components can render JSX, receive props, and embed JavaScript expressions inside of themselves.
- In a React application, most components should be stateless components. These are easy to test, debug, and they tend to be more reusable because they usually don't depend on how the rest of the application works.
- So far we've been using stateless components to create static, unchanging UIs. In a real application, we'd want to give some of our components more complex dynamic behaviors.

Stateful

- Stateful components aren't created using plain JavaScript functions, but with ES6 classes (which, if we want to get technical, are still JavaScript constructor functions once compiled).
- `state` is a special type of property attached to a class component that can contain data we want to associate with that component.
- Values stored on a component's state are different from regular variables because unlike regular variables, when a component updates its state the React application will update itself in the browser to reflect the change wherever necessary.
    - A component can set and update its own state, whereas its props are always received from up above and considered immutable (can't/shouldn't be changed).

Simply put, if you'd like your app to do anything – if you want interactivity, adding and deleting things, logging in and out – that will involve `state`.

Demo: Stateful Components

- See 13-Ins_BasicState
- http://localhost:3000/
- src/components/Counter.js
  - We create a new class named Counter which extends the React.Component class.
    - React.Component is a class built-in to React with special features unavailable to stateless components. Count inherits this extra functionality.
    - We set a state property with an object value with a count property set to 0.
      - Our component's state property must always be set to an object.
      - Because this component contains its own state, we call this a stateful component.
  - Note: render()
    - This method is built-in to React. Its job is to return the JSX that the component should render. Every class component needs to have this method defined

Demo: Stateful Components (cont'd)

- Note: The "Increment" button has an onClick prop `this.handleIncrement`
  - This is how a click event listener is defined in React.
  - Event names in React are similar to JS or jQuery, e.g. onClick, onSubmit, onChange, etc.
- Note: `handleIncrement` definition
  - Unlike render, this method is using arrow function syntax.
    - Why? The `this` wormhole.
    - We want our method bound to its parent, not the global
    - Inside `handleIncrement` we call `this.setState` and pass in an object
      - `setState` is built-in to all class components.
      - We ALWAYS use this.setState to update our component's state by passing it objects. Updating our state with this method tells our component that it should re-render itself and all of its children to account for the new state.

Takeaways

- We can use state to associate data with our components and keep track of any values we want to update in the UI when changed.
- We can define methods on a class component and pass them as props.
- The onClick prop can be used to set a click event listener to an element.

RTFM https://facebook.github.io/react/docs/react-component.html

Activity! Decrement Counter (10 min)

See 14-Stu_DecrementCounter

Review: Decrement Counter, Basic

We defined a handleDecrement method which decreases the counter by 1.

Review: Decrement Counter, Bonus

- We replaced the .panel-body div with a PanelBody JSX component.
- We pass the click count and the event listeners to the PanelBody component.
  - This component renders the same JSX that was removed from the Counter component. The only difference is that we're accessing the click counter and event handlers on the props argument.
  - Even though the buttons are inside of a child component, the Counter component's count state is still updated when the buttons are clicked.
  - When the count state is updated, the Counter component and any of its child components re-render themselves. This is what allows the view to be updated in the browser when the buttons are clicked.
- Even though data still technically only flows one way (from the top-down) in React, we can allow child components to update their parent's state by passing them a method created in the parent.

Bedtime Reading

- Forms
- Lifting State Up
- State and Lifecycle

References

https://daveceddia.com/visual-guide-to-state-in-react/

Further Reading

- https://medium.com/javascript-scene/jsx-looks-like-an-abomination-1c1ec351a918