ReactJS

Hunter Jorgensen

What is React?

React is an open-source JavaScript library (not a framework)

It was created in 2011 by an engineer at Facebook (Jordan Walke)

In 2013, Facebook released this tool as open source software

React Native, released in 2015, is a mobile framework that is used to develop apps on mobile platform (iOS, Android, Windows Phone, etc.) It uses React to make development more accessible to more developers

It uses a component view approach (think Web Components, but simplified!), which makes it super easy to manage logic, HTML and style all interchange

Pros/Cons of React

React is an extremely popular frontend library for a reason:

- It is easy and fast for developers to generate consistent and common templates (like classes, but for HTML/CSS)
- It intelligently manages updates to the DOM to be as efficient as possible

However:

- React can be quite a “heavy” library that can add latency to your website
- If not careful you can write less efficient code

Setting Up React

For assignment 2, you will be building a whole new app. Later on, you’ll want to setup a new Github repo and Heroku app/pipeline, but for now we’ll focus on setting up just the React part!

React uses npm, which we installed in our previous class

Setting Up React

There are many different ways to start a React app, but we'll follow the official recommendation from Facebook.

Go to the page where you want to create your react app and run the following:

```
npx create-react-app {{first-name--last-name-mini-assignment5}}

// Can also do `npm install -g create-react-app` if this doesn't work

// Then `create-react-app {{first-name--last-name-webdevspr2020-hw2}}`

cd {{first-name--last-name-webdevspr2020-hw2}}

npm start
```

That's it!

Finishing Up Setup

Next we'll want to clear out the src file:

`cd src`

`rm -rf *` (`del *` for Windows)

`touch index.js`

`touch index.css`

Add the following code to [index.js]()

Starting Variability

There are many ways to create a new React app, but our approach is the officially sanctioned practice suggested by Facebook.

The official approach to setting up an official React App is super easy, but allows for a fair bit of customization! You can optionally use tools such as Babel, Webpack, jest and others that are well suited for React or web development. I will will not discuss these libraries at a later point (great discussions topics, though!), or you can explore them on your own, but you are well suited to starting your first app.

Last Note on create-react-app

create-react-app is a great tool for understanding React, but it hides or abstracts a lot of configuring a new service so that you can focus on development. However, as you grow your app, you will likely want to change how the app works and runs or import libraries that are not included within the create-react-app script. In order to convert this app into a more open app, is a process known as ‘Ejecting’. We are not going to explore this here in this class, but it’s an important concept to be aware of.

Our First React App: Lessons to Look Out For

Before we move forward, we want to keep in mind the following ideas:

- React apps are made up of components
- Data flows from parents to children (props) or can be set within the component itself (state)
- JSX gives us a lot of good functionality
- There are 2 ways to make react components (Class vs Functional), with Functional components being highly recommended into the future

Functional Components

One shift in React in the last few years is the switch from class components to functional components. We'll explore class components in a later lecture.

React Components can be understood as functions (and functions in JS have a lot of complexity)

Functional Components

- React Functional Components are components written as function
- The return statement is what is rendered
- You can use arrow or formal functional format
- Props as passed in as “arguments” to the function
- We can use “hooks” to handle more complex actions (discussed shortly)
- Is increasingly the more common way of writing a component

What are Hooks? And why?

In old React, a component will update the view when its state is changes or new props are passed in, and there are functions that exist when these changes occur.

But with functions, there’s not explicit way to set a state, or perform logic when a state change occurs. This is where Hooks come in.

useState

The most common React Hook is useState. The sets and maintains the state of a piece of data:

const [state, setState] = useState(initialState);

(Note that this is using the array destructuring syntax)

state is a variable that represents the current value

setState is a function that allows you to update the state

useState is a function provided by React to generate these 2 variables

initialState is the state when your component first loads

useEffect

useEffect is the SECOND most common Hook and what it means is that whenever the current component changes, or a particular piece of that component changes, some side “effect” should happen.

useEffect(() => console.log(“print every time the DOM changes”);

or

useEffect(() => console.log(“print every time a value updates”, [value1, value2]);

useReducer

You may recognize the “reducer” idea from Redux. Here the function takes an initial state and a reducer function that modifies the state via a provided dispatch function:

const [state, dispatch] = useReducer(reducer, initialState, initFunc //optional//);

reducer is a function that will modify the state

dispatch is a function that is called when you want to update the state

How does this compare to useState?

useContext

With the arrival of Hooks, React has offered a built in alternative to Redux known as Context. The primary purpose of context is to share data across multiple components easily.

useContext and useReducer are hooks that behave similarly to their Redux cousins.

useContext accepts context and returns the current state and a dispatch method. We’ll see useContext in action when we learn about Redux.

Review: Definitions

Component - A custom and reusable element (similar to HTML)

Props - In React, data that is passed into a component (similar to an attribute in an HTML element)

State - In React, data that exists in an element

Extra Credit

Go into your code for mini assignment 3 and update that into a React app. Take an image, push it to a new repo and share the image and repo with Nandish for 3 extra points by Saturday night.