Script

Pearce:

- React, which also goes by the name of ReactJS, is a javascript framework used to create sophisticated user interfaces for webapps
- React was initially launched by Facebook and then made open source. Facebook needed a way to make more complex interfaces that loaded quickly.
- React allows us to write javascript and html in the same file, effectively consolidating our frontend code.
- As you develop your app, it will change in real time as you compile and test your content, allowing you to see changes in real time which speeds up development time

Next slide

- React has many benefits to using standard, vanilla html and javascript.
- Its main advantage is that React is able to be easily modularized. This allows pieces of your frontend code to be very easily reused in a similar way you would pass in new parameters to a function. It also allows many developers to be working on the same frontend code at the same time.
- React has a massive library of support with other libraries and has easy integration with the integration layer of an application and the backend.
- React also has a vast open source community, lots of documentation line, and many people willing to support you in switching your frontend to React.
- Some companies that use React are Instagram, Netflix, New York Times, Airbnb, and Discord

Next Slide

- A core aspect of React is the use of the Virtual DOM.
- DOM stands for Document Object Model
- Examples are Div, Section, Table, and Figure
- Table follows a tree structure with a clear parent child relationship
- Table followed by table row which contains table data
- All can be used with Ract.js

Next Slide

- The difference is that React.js uses the Virtual DOM
- The Virtual DOM

An analogy people use to describe the Virtual DOM is when you're cleaning something in your room, you don't clean the entire house - you just pick up the object.

Map dependencies between components and then only update specific components when one of their dependencies changes because of the parent child dynamic

Sebastian:

- Now we'll talk a bit about the specifics of coding in React.
- As mentioned earlier, React is modularized. Individual modules are called Components or sometimes functions if you use a different design model.
- You can think of React components as pieces of HTML code that have information and logic tied to them.
- The logic happens in the render piece of the code and is coded in basic JavaScript. In the example provided, this is where the console log statement is.
- Whenever a dependency relationship changes in the Virtual DOM as mentioned previously, the render statement is re-run.
- This is important to differentiate from the "return." A return statement will return JSX elements which are very similar to normal HTML but can also include JavaScript injections handled in your render statement very easily. In the code snippet, the div element is in the return statement and acts as normal HTML.

Next slide:

- As mentioned previously, React is modularized. When you define a component, you can duplicate it very easily as shown below

- The App class serves as a parent component for the Button class and instantiates the Button class twice. Instead of writing the button code twice, we can just call it like a function.

Bryce:

- Now we've discussed modularization and the basics of coding in React, but we haven't touched on the flow of data yet.
- In vanilla javascript, communication between different areas of your frontend applications is pretty self explanatory, but in React, the communication between components needs to be explicitly specified so that the Virtual DOM can work. When a component of our application changes, we need these changes to reflect in other specific areas of our frontend, but we don't want it to load the entire web page like in vanilla javascript.
- Components communicate with each other in React in two ways - props and state.
- Props is used when static data is passed from a parent component to a child component that isn't meant to change. You can think of this as parameters passed into a function. State is used when information has the ability to change. We can have this information change within its own component or even have the changes propagated throughout the Virtual DOM to other components.

Next slide:

- For an example of props, we pass two different texts; "hello 1" and "hello 2" into the props of a Child component that then renders those props in an h1 element.
- You can see this in the web page to the bottom right

Next slide:

- For an example of state, we have a component which renders a name. The name is initialized as "Andrew", but when you click on the header tag, it updates to "Jack". The component does this through a callback function which when called re-renders the component and updates it to the frontend to show the new value.

Next slide:

- In a slightly more complex example, we have two components which are connected. The component Button is a child component of the App component and receives App's state of clickEvent in its props. When a button is clicked in the child component, the callback function in App is propagated and updates the state in its parent component. Whenever we click on the button, data in an entirely separate component is updated. This is extremely useful for propagating dynamic data throughout the entire application.

Next slide:

- React has some similar concepts to 3155. For example, it relies on higher order functions and callbacks as we mentioned, and also has abstract data types of type Class, Child, etc.

- We'll go into more specifics on a few of these similarities.
- Callback functions, which are pervasive in React and essential to how data flow works.
- Callback functions are functions which are declared to "run at some time in the future".
- React, and Javascript, works on a Synchronous and Asynchronous clock that has many different components working together and many tasks that have effects which need to be propagated after some action has occurred.
- In the example below, we have a Button class component that is instantiated by the App component
- The app component has a callback function called uncover that is bound to its modularized state, but instantiated by its child component - the Button component
- When someone clicks on a button, an event that could happen anytime in the future, the callback function is instantiated in the App component and a line of text is uncovered.
- When the button is clicked again, the same callback is instantiated and the button disappears again. We keep track of the visibility of the button's css through state used in the component and update it so it's automatically reflected in the frontend

Next slide

- Here's a quick video demonstration of the previous code running. Whenever the button is pressed, the code disappears.

Next slide

- Another thing React relies upon heavily is Abstract Data Types.
- All components in React are ADT's, defined by a set of logic as we've shown and able to be instantiated anywhere in the codebase.