**REACT**

What is ReactJS?

React is a js library for building user inter faces(UIS). Some people uses it as the V in MVC. Building large applications with data that changes over time.

* Library for creating interfaces
* Focuses on the view
* Uses a virtual DOM
* Done through component
* One way data flow

Angular One and other frameworks, you create a two-way relationship between the model and the view. React can do the same things but keep the flow of data going one way. Now the reason that's good is because you end up managing everything through the main component state.

* States
* Props

If something changes in your State or in your data, then React with re-render the DOM for you automatically. All you have to worry about is what happens to your data. The main component passes information to sub-components via something called Props, or properties. Think of them as properties in an HTML tag, like the anchor tag has an HREF tag and tags like image require you to have like a source attribute. Props, or properties, can both pass information down to sub-modules but also trigger actions in the main module.

* Jsx – combination of javascript and XML
* **Component** in React works similar to JS functions: It generates an output every time it is invoked. Components in React are JS classes that inherit from the React.Component base class.
* The **Virtual DOM** is an in-memory representation of real DOM elements generated by React components before any changes are made to the page.
* **Virtual DOM diffing** allows React to minimize changes to the DOM as a result of user actions – therefore increasing browser performance.
* **ReactDOM** is used to render components to our HTML pages as it reads output from a supplied React Component and adds it to the DOM.
* **JSX –** The markup we use for React apps , It looks similar to HTML, but ultimately gets transpiled to JS function calls, which react will know how to render to the page.
* **Map function:** native js function

Need react lib – to use components

React-dom - to render components

The DOM, or document object model, is the structure of HTML elements that make up a webpage. It also refers to the API for how these page elements are accessed and changed. Reading and writing to the DOM is slow. Reading and writing to JavaScript objects is faster. Every time we read and write to the DOM it's expensive. It really starts to affect our application's speed. React has a virtual DOM that writes to the browser DOM only when it needs to.

Components are small user interface elements that display data as it changes over time. Then these components are composed together, nested inside of one another to create entire user interfaces. Websites can use React for all of their user interface or just bits and pieces of it. I'm gonna take a look in the next tab at Nordstrom.com. Nordstrom uses a lot of React, as we can see here in the React Detector.

A stateless functional component is just a simple function that returns react elements

Sending properties to a component is very similar to adding attributes to HTML. .

The component lifecycle provides hooks for creation, lifetime, and teardown of components. These methods allow you to do things like add libraries, load data, and more at very specific times. The mountain lifecycle has several methods that we've used already. For example, getInitialState is going to be called once and will set the default for a state. componentWillMount is called right before the render, and it's the last chance to effect state prior to the render. The render method we know well, this is the only required method, and componentDidMount is going tofire right after the render, so after a successful render, we can now access the dom, the component has been rendered, and now the user can interact with it.

The component lifecycle also provides methods for updating. componentWillReceiveProps, once this method is called,we get the opportunity to change the object and effect state. shouldComponentUpdate and componentWillUpdate are invoked right before rendering, and are often used for optimization. We're only going to call these methods if something has changed. Finally, we'll have the render method here again, and that's going to be part of the updating lifecycle as well, and componentDidUpdate is going to fire right after everything in the dom has been updated.

omponentWillUnmount is called right before the component is unmounted. This can help us do things like clean up dom elements and invalidate timers. So when componentWillUnmount is called on the parent, all of the children are unmounted as well. The component lifecycle has many different methods that you can use to optimize your applications. In the following videos, we're going to go over demos that explain these steps, and then we're going to add some more robust lifecycle functions to our note application.

Components in React

* Create with React –
  + createClass(object spec) - () – to determine what the component does
  + Takes Object
  + Needs render() method - sets up what will be sent to the virtual DOM
  + Jsx syntax –js and xml
* Rendering to DOM
  + Use ReactDOM’s

Render(reatElement, DOMElement)

* + Object to Render
  + DOM Element to Target

creatClass

method – render – sets up what will be sent to the virtual DOM

React is a library for creating user interfaces and that means that it's not necessarily concerned with the whole process of creating an application.

React is *fast*. Apps made in React can handle complex updates and still feel quick and responsive.

React is *modular*. Instead of writing large, dense files of code, you can write many smaller, reusable files. React's modularity can be a beautiful solution to JavaScript's [maintainability problems](https://en.wikipedia.org/wiki/Spaghetti_code).

Components:

React is *scalable*. Large programs that display a lot of changing data are where React performs best.

React is *flexible*. You can use React for interesting projects that have nothing to do with making a web app. People are still figuring out React's potential. [There's room to explore.](https://medium.mybridge.co/22-amazing-open-source-react-projects-cb8230ec719f#.o5umedb6v)

React is *popular*. While this reason has admittedly little to do with React's quality, the truth is that understanding React will make you more employable.

JSX elements are treated as JavaScript expressions. They can go anywhere that JavaScript expressions can go.

ReactDOM is the name of a JavaScript library. This library contains several React-specific methods, all of which deal with [the DOM](http://www.w3schools.com/js/js_htmldom.asp) in some way or another.

ReactDOM.render is the most common way to renderJSX. It takes a JSX expression, creates a corresponding tree of DOM nodes, and adds that tree to the DOM. That is the way to make a JSX expression appear onscreen.

2. What is the significance of the React.createClass() method?

- Creates a component

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

- Displays the components or the HTML you generate onto the page

4. What is JSX? Why does React recommend it? (Hint: See documentation on "Displaying Data")

*JSX* is a syntax extension for JavaScript. It was written to be used with React. JSX code looks a lot like HTML.

- It is Javascript XML

- And it allows you to treat HTML content as being totally Javascript developed. It bundles HTML and Javascript

5. How do I incorporate a JSX script into my HTML? (Hint: See documentation on "Displaying Data")

- Include a <text/babel> script (this compiles the vanilla javascript)

6. What is Babel? And what role does it play in converting JSX into vanilla Javascript?

- Babel is a transpiler that converts JSX into plain Javascript. It allows us to create more advanced Javascript in easy syntax.

7. What is the significance of { } curly braces in JSX? (Hint: See "JSX in Depth")

- {} braces convert into Javascript variables

8. What is the difference between a prop and a state? (Hint: See articles on "Thinking in React", "Interactivity and Dynamic UIs")

- Prop is a static unchanging aspect to a component. (the "name" of the "chat box")

- State is constantly changing (whether chat is open or closed)

9. Create a Simple Hello World Application with React

Part II:

- Note at least 1-3 questions, curiosities, or mysteries you stumble into as you tread through the ReactJS documentation

Part III:

- If you finish early, begin reading and working through the ReactJS Tutorial: https://facebook.github.io/react/docs/tutorial.html

**Jquery Madness**

* Without an organization structure, code quickly become a series of erratic DOM Manipulations

React – uses a virtual access that a middle man with

**React Goodness**

* In contrast, React utilizes a state or Virtual DOM that acts as a middle man with pre-defined rules for how each component will behave.

Concepts of Components:

Layout and logic bundled together. More easily tested and more reusability. They work similarly to js functions. They ultimately generate html code. They are JS classes that inherit from React.Component base class.

Jquery – primary use is DOM manipulation.

Virtual DOM: is in memory rep of real DOM elements generated by react components before any changes are made to the page.

Vitrtual DOM Diffing : allows React to minimize changes to the DOM as a result of user actions – increasing browser performance

Bable Is a tool that converts jsx into js.

Install babale in sublime

* <https://packagecontrol.io/installation#st3>
* Paste it into sublime console

Webpack Installation

Npm i webpack –g --- globally installing

It is a bundler. It groups all the files together and outputs a single file

Webpack.config.js

// This code will be compiled by webpack according to the babel specifications -- entry: "./app/app.js"

// The plain compiled Javascript will be output into this file output: {filename:"public/bundle.js"},

// This will be what we loader: 'babel', -- that will look at source code and coneverts jsx to js

// These are the specific transformations we'll be using. ------- presets: ['react', 'es2015']

To run Examples

19.1 -02

1. 1: Run npm i 2: Run webpack -w in one window & 3: Run node static.js in another window

To generate bundler ---- Webpack –bundler

19.1- 05

{/\*Inserted the variables and simple calculations using curly brackets \*/}

<h2>My name is {name}. But you can call me...</h2><h1>The JSX Boss!</h1><hr />

<h2>I can do math: {num1 + num2}.</h2><h2>I can generate random numbers: {Math.floor(Math.random() \* 10) + 1 }, {Math.floor(Math.random() \* 10) + 1 }, {Math.floor(Math.random() \* 10) + 1 }.</h2><h2>I can even reverse my name: {name.split("").reverse()}</h2>

19.1 -06

// Here we create a variable for holding the name and birthday

var dob = moment("1989-02-14", "YYYY-MM-DD"); <h2>That makes me: {dob.fromNow(true)} old.</h2>