February 24th, 2018 –

REACT_JS [CODECADEMY]

JAVASCRIPT LIBRARY DEVELOPED AT FACEBOOK
OPEN SOURCE PROJECTS

REACT IS FAST – COMPLEX UPDATES QUICKLY
REACT IS MODULAR – MANY SMALLER, REUSABLE FILES
REACT IS SCALABLE – BEST USED DISPLAYING CHANGING DATA
REACT IS FLEXIBLE – POTENTIAL STILL UNKNOWN
REACT IS POPULAR – HELPS TO BECOME EMPLOYABLE

1. WHAT IS JSX

- A. A syntax extension for JavaScript. Written to be used with React (looks a bit like HTML)
    - A. This means JSX is not valid JavaScript and must be compiled and translated to JavaScript before reaching a web browser
- B. Basic unit of JSX is called a JSX element
    - A. Example: <h1>Hello World</h1> looks like HTML, but in a .js file
    - B. JSX element treated like JavaScript expression in that it can be:
        1. *Saved in a variable*
        2. *Passed to a function*
        3. *Stored in an object or array*
            - a. *const navBar = <nav>thing goes here</nav>;*
            - b. *const myTeam = { center: <li>Tim</li>, pointGuard: <li>Jim</li>, … };*
        4. *Etc.*
- C. JSX elements can have attributes
    - A. Looks like HTML element (can have one or multiple)
        - a. *const navBar = <nav id="nav-bar">thing goes here</nav>;*
- D. Nested JSX
    - A. To make it readable use HTML-style line breaks and indentation
    - B. If expression takes up more than one line, then you must wrap the multi-line JSX expression in parenthesis
    - C. Can be saved as variables, passed to functions, etc.

1. *const nestedExample = (*

```
    <a href="link here">
        <h1> Click link </h1>
    </a>
);
```

- D. JSX Outer Elements
  1. *A JSX expression must have exactly one outermost element*
     - a. *i.e. the first and closing tag of a JSX expression must be the same*
     - b. *You can always just wrap it in a <div> if this is an issue*

E. Rendering JSX - Make it appear on the screen

- A. ReactDom
  1. *Name of the JavaScript library that deal with the DOM*
- B. ReactDOM.render()
  1. *Most common way to render JSX*
     - a. *Only updates DOM elements that have changed (called "diffing")*
       - i. React is so successful because of this significant ability
       - ii. Accomplishes this because of *the virtual DOM*
         1. Entire Virtual DOM gets updated
         2. Virtual DOM is compared to snapshot of DOM right before the update
         3. React figures out which objects have changed and change only those objects in the real DOM
         4. Changes on the real DOM cause the screen
  2. *Takes the JSX expression, creates corresponding tree DOM nodes, and adds that tree to the DOM*
  3. *The first argument (HTML looking thing) being passed should evaluate to a JSX expression, and it will be rendered on the screen*
     - a. *It doesn't have to literally be a JSX expression*
     - b. *It could be a variable as long as it evaluates to a JSX expression*
  4. *The second argument tells where to put the first argument on the screen*
     - a. *Example: document.getElementById('app')*
     - b. *Note: The first argument is appended to whatever element is selected by the second argument*

2. ADVANCED JSX

A. Grammar in JSX is mostly the same as HTML with subtle differences
   1. *class vs className*
      a. *class in HTML is className in JSX because class is a reserved word in JS which JSX get translated you can't use class*
         i. JSX className attribute automatically render as class attributes
   2. *Self-Closing Tags*
      a. *Must include the / in self closing tags with JSX (optional in HTML)*
         i. <br /> is JSX is ok but <br> is not (even tho both ok in HTML)

B. JavaScript in JSX (which is in JavaScript file)
   1. *Wrap in { } for JSX code to be read as JavaScript*
      a. *Example: <h1>{2 + 3}</h1> will show 5 but without the { } it will literally show 2 + 3*
   2. *Injected JavaScript is part of same environment as rest of file so you can access variables inside of JSX expressions even if variable declared outside*
   3. *Object properties are often used to set attributes (organize code)*
   4. *Event Listeners (valid event names)*
      a. *Attribute value should be a valid/defined function*
      b. *Written in camelCase for JSX not all lowercase like HTML*
   5. *Conditionals: If statements that don't work (can't use an ' if ' in JSX)*
      a. *Explained here*
      b. *Common to keep the if else outside of JSX tags, not injected between*
      c. *Ternary Operator – more compact way to write conditionals*
         i. Explanation: x ? y : z (if x truth return y, if x false return z)
      d. *&& operator*
         i. Works best in conditionals that will sometimes do an action but other times do nothing at all
      e. *.map()*
         i. Is best bet for creating lists in JSX for example:
            1.
```
const arrays = ['thing1', 'thing2', 'thing3'];
const listArray = arrays.map( arrayItem =>
        <li>{arrayItem} </li>);
ReactDom.render(<ul>{listArray}</ul>, document.get … );
```
      f. *Keys – JSX attribute and the value should be unique (like and id)*
         i. React uses them internally (don't see it) to track lists
         ii. React might scramble lists if you don't use keys correctly

iii. Needs keys if either of the following is true:

1. The list-items have ‘memory’ from one render to the next
    a. *i.e. was something checked off a list?*
2. A list’s order might be shuffled
    a. *i.e. maybe a lists search results*
3. Otherwise you don’t have to use keys (but doesn’t hurt if you do)

C. React.createElement

1. *You can write React code without using JSX (majority of programmers do use JSX, but don’t have to)*
    a. *Example in JSX*
        i. const title = <h1>Hello World</h1>
    b. *Example of React without JSX*
        i.
```
Const title = React.createElement(
   “h1”,
    null,
    “Hello World”
);
```
    c. *When a JSX element is compiled the compiler transforms the JSX into the method above*

3. REACT COMPONENTS