Section 3 react basics/components

* **Getting started**
  + Index.js holds code that gets run initially.. This is just something you have to know
    - “Transformed version of that code”
    - There are scripts running behind the sidelines that transform
  + React is still JS but there are certain things that would not wokr in regular javascript file
  + Uses single html file… central place react interface is rendered into
    - Html will contain a central Div where react interface is rendered.. Called root in the started file
    - For some reason you omit .js for other js files
  + **Jsx**
  + Special syntax developed by react
  + Html inside JS?
    - Works because of npm process that runs transformations behind the scene
    - with react you build yourown custom html elements or components
    - Specify desired target and DOM manipulation happens behind sidelines…declarative approach
  + Can create function to create HTML components,, in vanilla JS it’s called an imperative approach as you have to specify elements, create them, assign contents, then append to specific elements
  + **Creating custom components**
    - Best practice to put new components in new files
      * Normal to end up with tons of files
      * Good to setup a dedicated components folder
      * In this file index.js is the starter file, then app.js is root file… all other components will be nested within app.js root file, or somewhere else
        + With react you build a component tree
    - Components in react are just special JS functions that returns JSX code or html elements
      * Create new components then export and import/nest in a different file,,, often root file (app.js)..
        + Then you can use the imported component however you please as if its an html element… easy
        + Components have to be first letter uppercase as components used as they are interpreted as html if it begins with lowercase
        + Create component, export it, import in file where you want to use it, then it functions like an html element
    - **More about JSX code**
      * Must only have one root element per return statement
        + Can wrap all in a div wrapper… also good to wrap in brackets so JS knows its a single element
      * Can create css file and import to root or other JS files… when adding clases to js you use className instead of class
        + With new syntax it is important to know that JSX looks like html, but isn’t so certain things have to be different… className is necessary because class is a reserved word in JS
      * In jsx can run JS expressions and use consts inside the seemingly html elements.. Separate with {}
    - **Props** 
      * Way to make reusable components passable with different data
      * Better to store actual values as properties in the root file as opposed to the properties file
      * Can save info in object on main file then use attributes with the REACT {} syntax to pull from them
      * Makes easier to pass data between components
    - **Splitting components**
      * Components can become very large
        + Component based framework makes it easier to split these further..

Can take a lengthy component and split into multiple components

* + - **Sum**
      * Components are the building blocks of react.. And they are made up of HTML or JSX code, javascript components and props that allow you to pass info between them.
    - **Composition**
    - Process of building in smaller components to create complex UI is called composition
      * New part of “composition”
        + Component that serves as a shell around any other content
        + Components we have so far output very specific elements that are configured through props.. Sometimes you might create components that act differently
        + Common to have components where you don’t pass everything thru props but instead thru opening and closing tags.
        + Can write css code in a component– typical to create div iwht style sheet uploaded then add the class as an attribute to the DIV,,, but can also create a component with built in style then use that in place of the div so the style is built into the component and used as an opening/closing tag..
    - **Wrapper components**
      * + Custom components cant be used as wrappers for other components out of the box though– but there is a solution with props but not with attributes… a built in component that every component autmaticcalt receives
      * **Children props**
        + Children will always be the content between the opening/closing tags of your custom components

function Card() {

return <div className="card">{props.children}</div>

}

export default Card;

<Card className="expense-item">

<ExpenseDate date={props.date} />

<div className="expense-item\_\_description">

<h2>{props.title}</h2>

<div className="expense-item\_\_price">${props.amount}</div>

</div >

</Card >

* + - Another issue with using custom components this way is that classes imported from external .css sheets won’t work out of the box without a workaround.
      * In this ex. ^ css style is built into the component, but other style is also added on, but it won’t work as-is
      * Common to see complex custom wrapper components with things like modals or alerts.. Being able to extract allows you to extract that and keep other components clean
* **More on JSX**
  + Jsx code is syntatic sugar that is usable by react thanks to behind the scenes transformations, but not recognizable on its own by the browser.
  + Reactdom(in the json folder) is integrated in the initial index.js file.. But theres another file called react which is an object that is integrated behind the sidelines.. You could create elements using React to work around JSX but jsx is more readable and nicer to look at.
* **Alternative function syntax**
  + Arrow functions can work