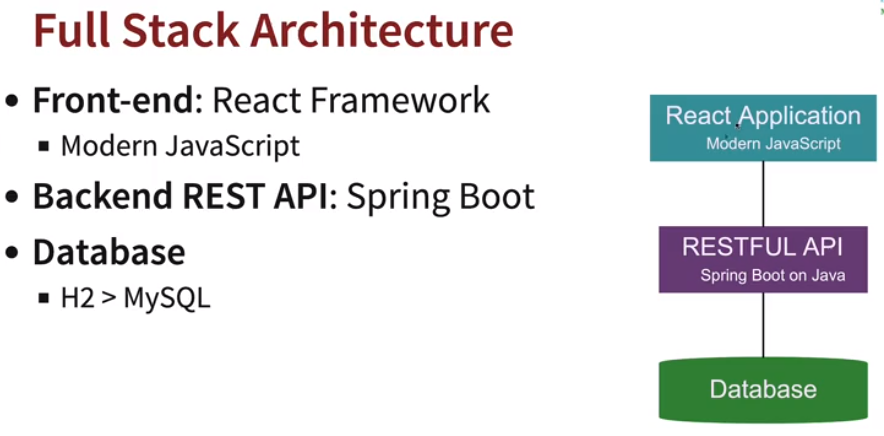
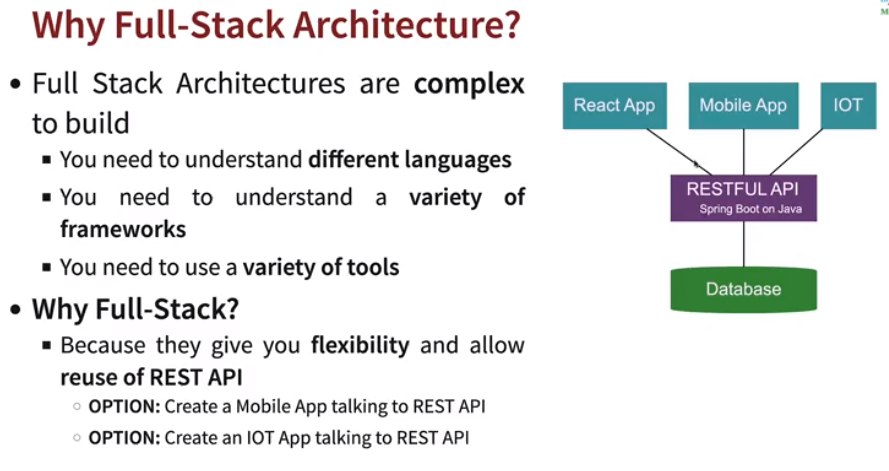
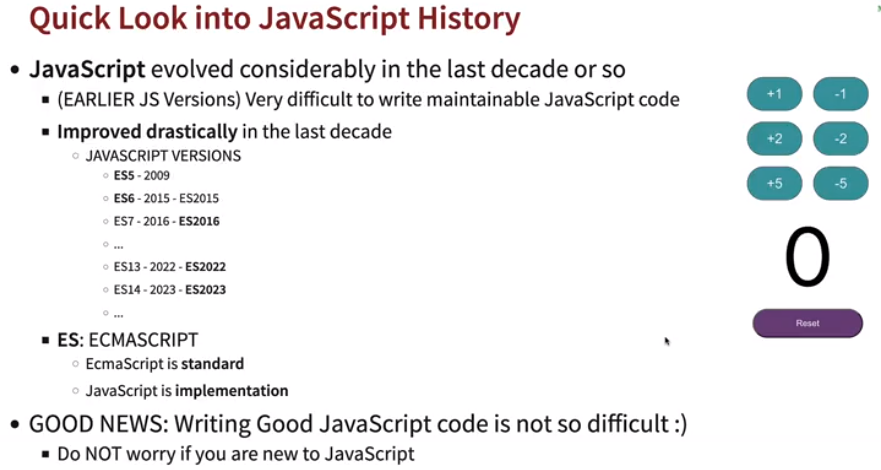
**FULL STACK**

* Full Stack development is complex but it allows us to connect different types of applications, like webapplication, RESTApi and IOT applications by reusing the backend.
* It has mainly 3 components Application(React) 🡪 Restful API 🡪 Database.

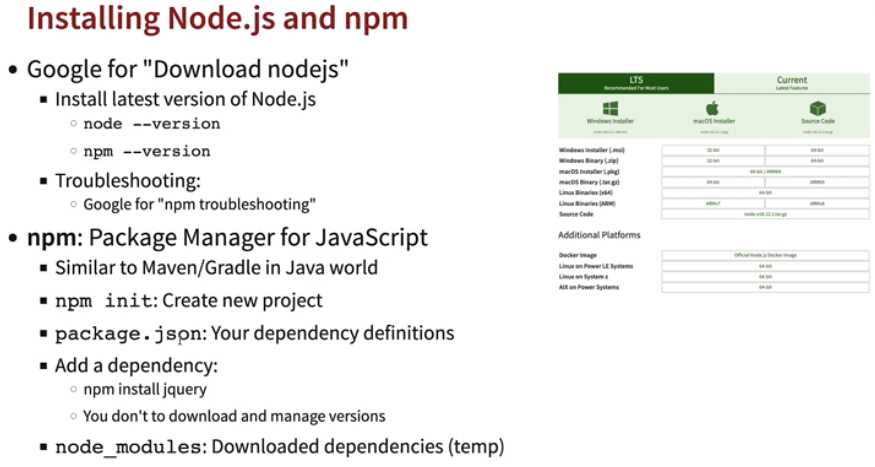
****

****

* **JavaScript:** Earlier writing JavaScript was difficult using DOM (document object model), i.e., manipulating different JavaScript pages.
  + **ES:** ES stands for ECMA Script which is like an interface for JavaScript or standard.
  + **JavaScript:** JS is implementation of ES (ECMA Script).

****

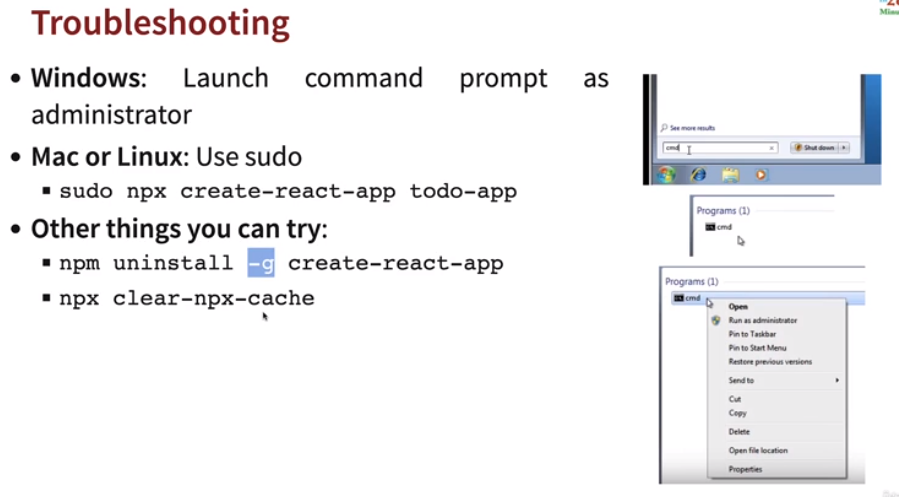
* **Installing Node JS:** <https://nodejs.org/en/download>
* **Node JS:** Node JS is an open-source, cross-platform JavaScript runtime environment and library for running web applications outside the client's browser. Just like JRE for the Java programs.
* **NPM:** Node Package Manager 🡪 It's a library and registry for JavaScript software packages. Similar to Maven or Gradle. It is used install, update and delete packages.
* **NPX:** Node Package Executor 🡪 It executes JS packages without installing it.
* **Create new project:** In command prompt type: *npm init.* It will initialize the project and create a package.json, similar to pom.xml which consists of all dependencies definitions.
* **node\_modules:** It is a temp folder where all downloaded dependencies are kept.

****

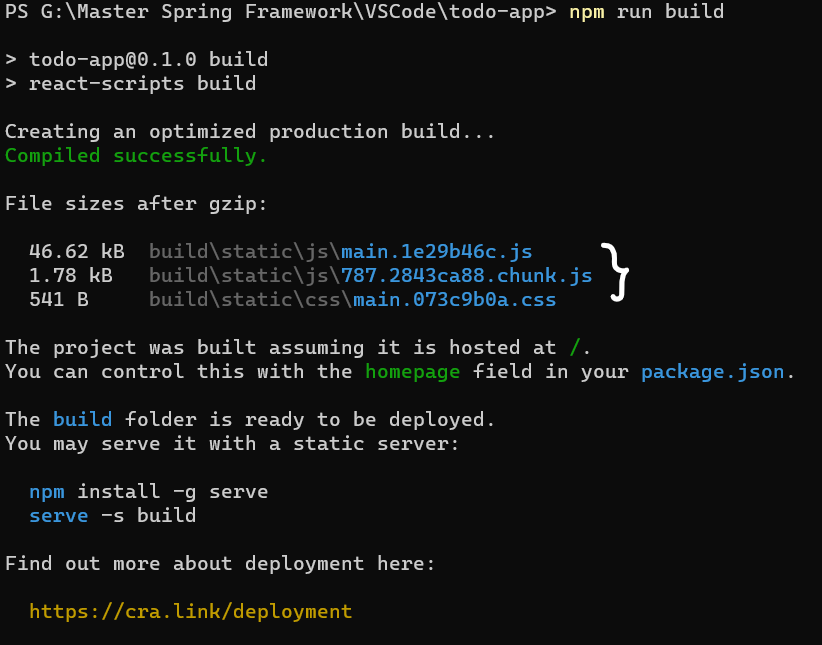
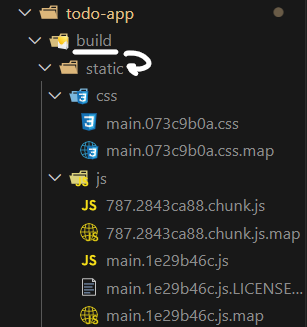
* **React JS:** React JS is used to create Single Page Applications (SPA), which is compatible with Windows, Mac and Linux OS.

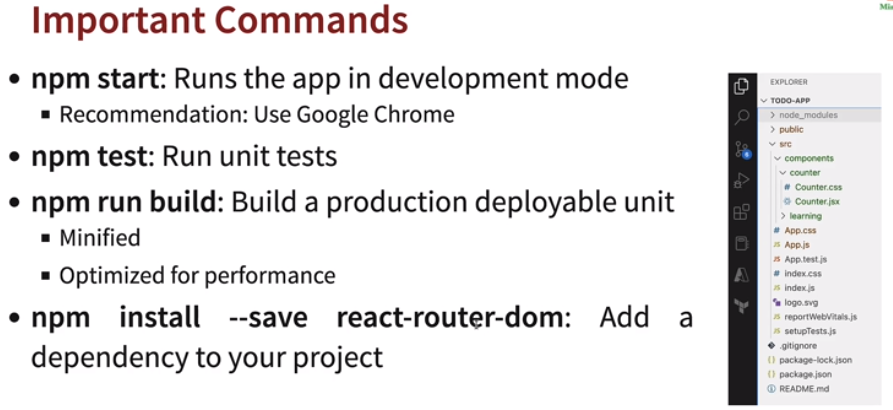
****

****

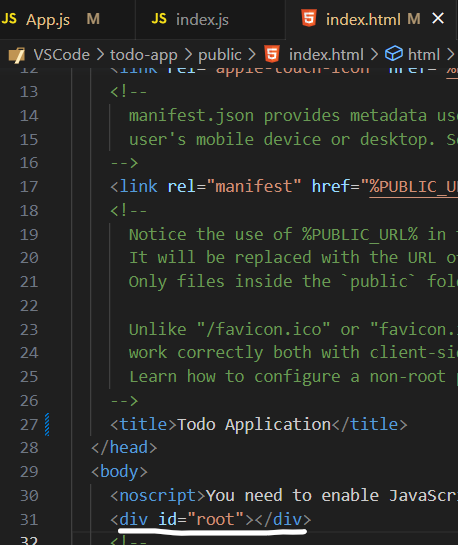
****

* **NPM Commands:** 
  + **npm start:** To start your application in development mode. The changes made in application files will immediately reflect in browser.
  + **npm test:** To run Junit test or test cases defined in application.
  + **npm run build:** Similar to like creating jars for java, it creates minified version of your application code and build files are necessary enough to run application in any environment.
  + **npm install –save <dependency>:** Add a dependency in project.

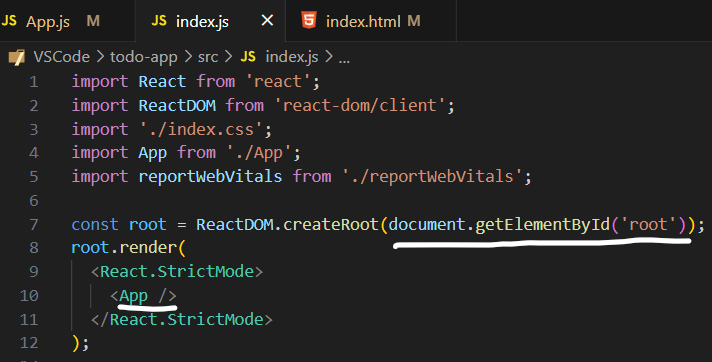
** **

****

* **Folder Structure:** 
  + **Index.html:** It contains a root div which is then mapped as a component in the index.js.

****

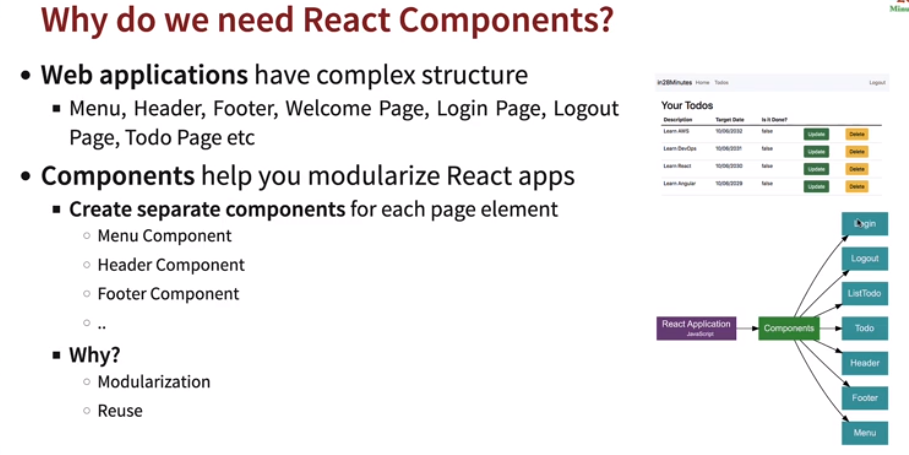
* + **Index.js:** It renders the app (root) component. App component will be the first to be loaded in.

****

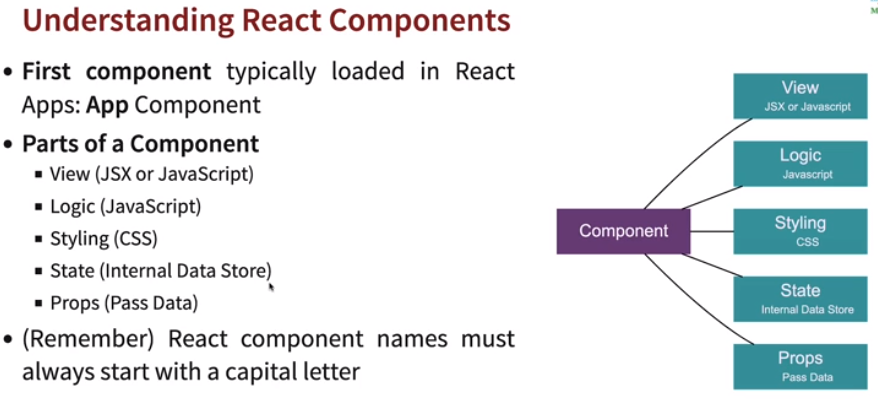
* + **App.js:** It contains the code for app component.

****

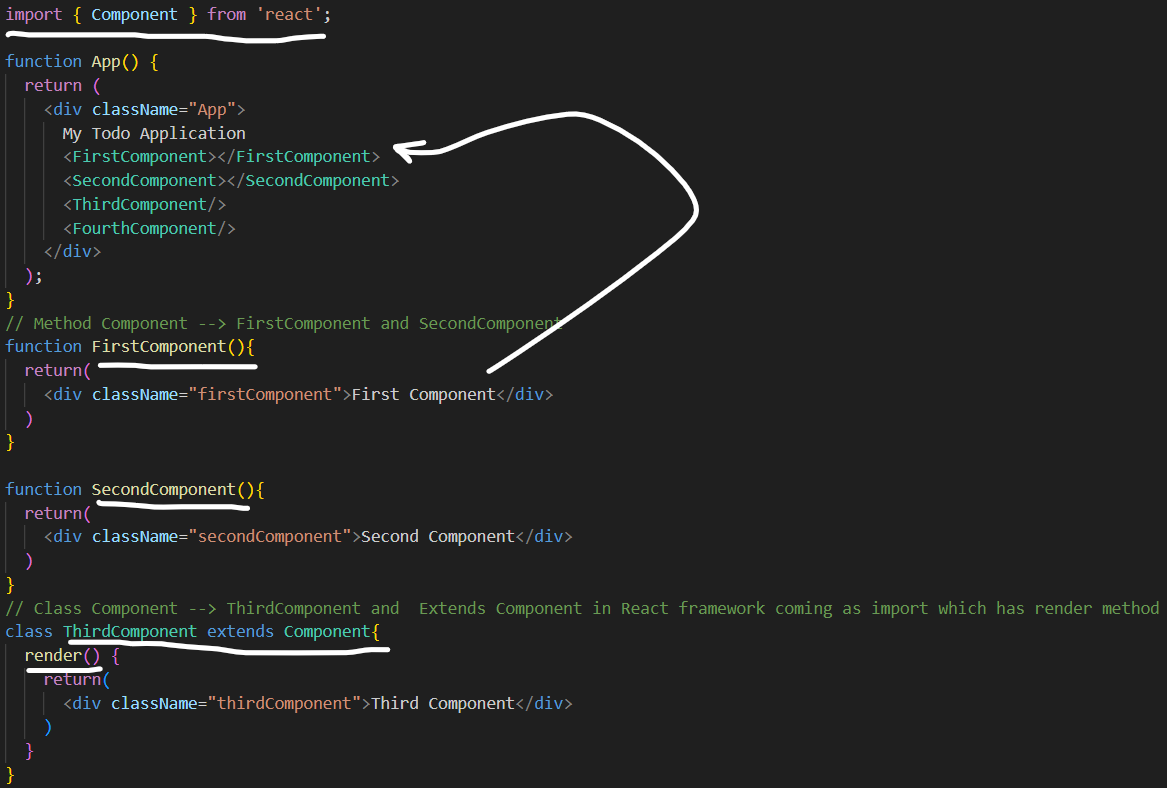
* **React Components:** React components helps in modularizing (separating into different sections) application. Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render().

****

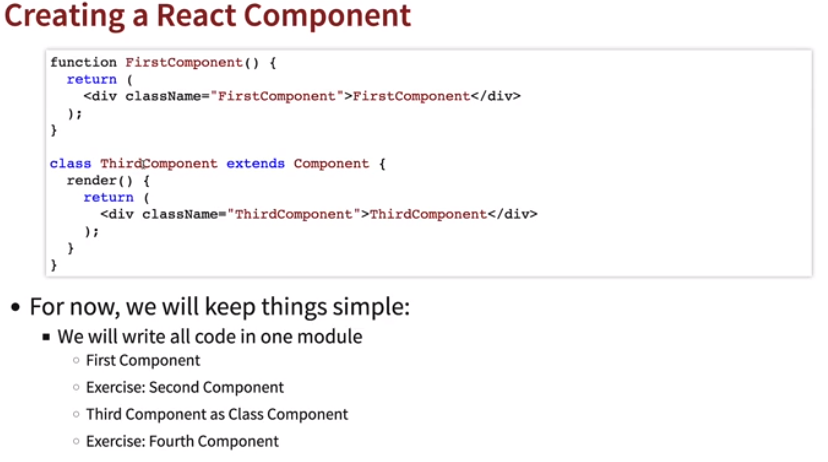
* **First Component loaded in React?** 
  + The first component that is loading in application is *App* component.
  + Components must start with capital letters.
  + All other components created will be child of *App* component.
  + A component can have:-
    - View (JSX or JS)
    - Logic (JS)
    - Styling (CSS)
    - State (Internal Data Store)
    - Props (Pass Data)

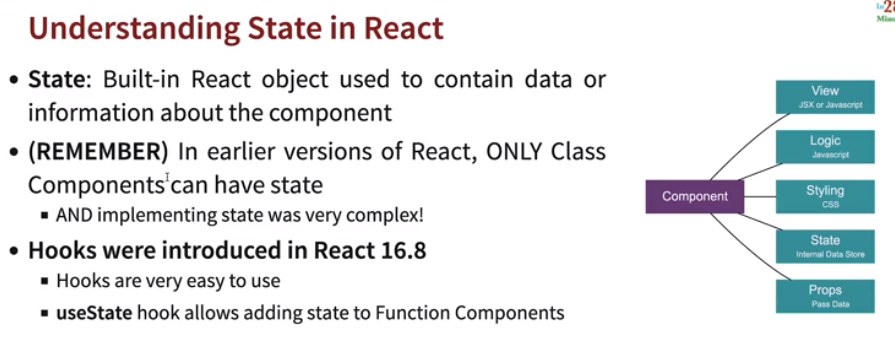
****

* **Creating a component:** Every component should be a child of *App* component or called in *App* component.

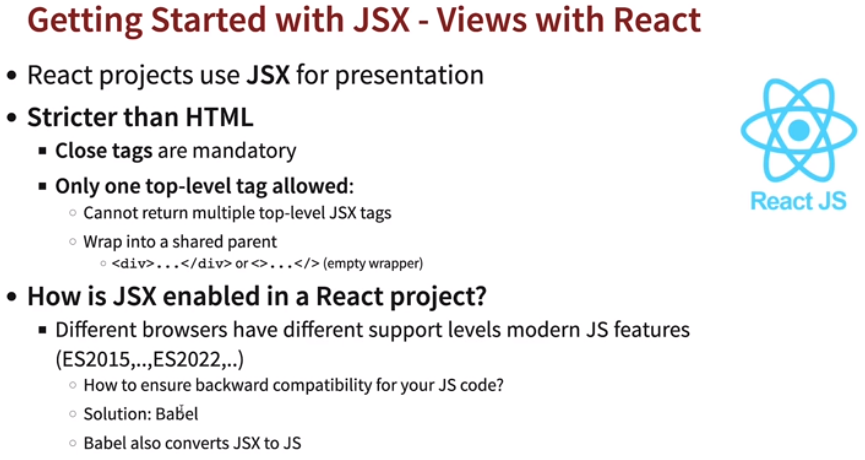
****

* In earlier versions of *React < 16.8* only *Class Components had the state* (some data) whereas onwards *React >= 16.8 Function Components also have states* as *Hooks* were introduced.

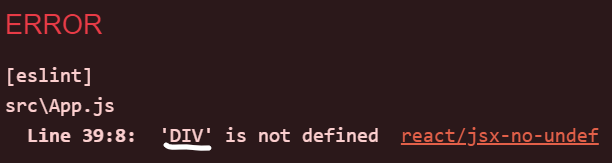
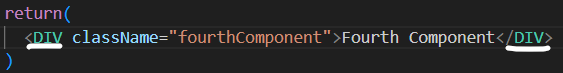
****

****

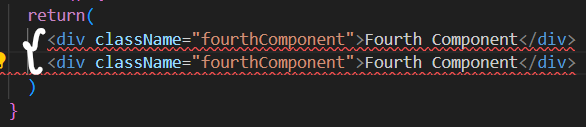
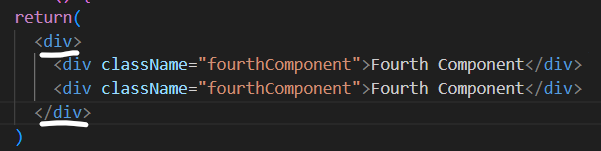
* **JSX:** JSX stands for JavaScript XML. JSX allows us to write HTML in React. JSX is an extension of the JavaScript language based on ES6, and is translated into regular JavaScript at runtime.

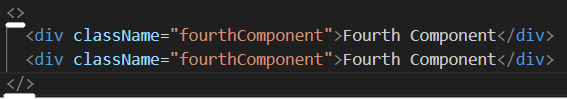
****

* + Stricter than HTML. Case of components should be capital case or elements needs to be small case.

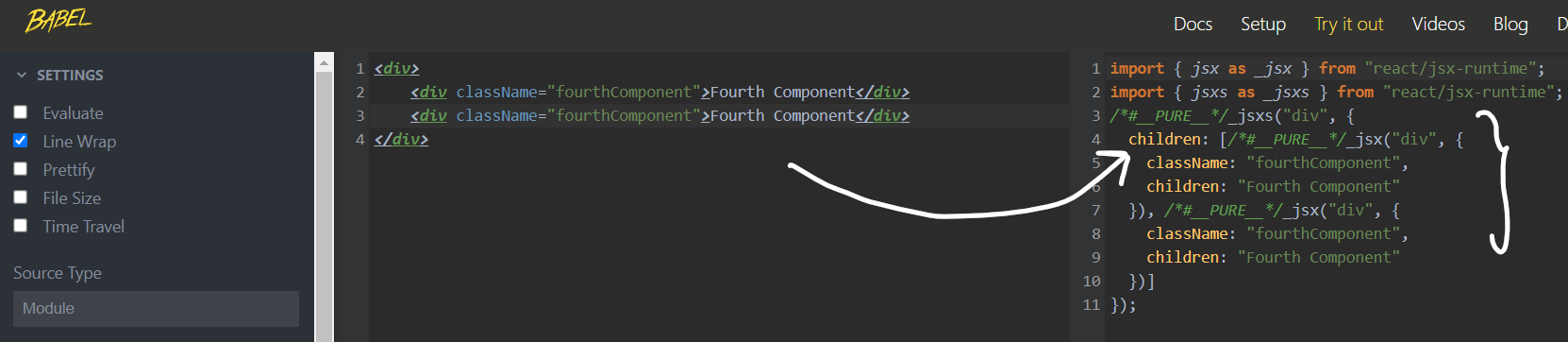
** **

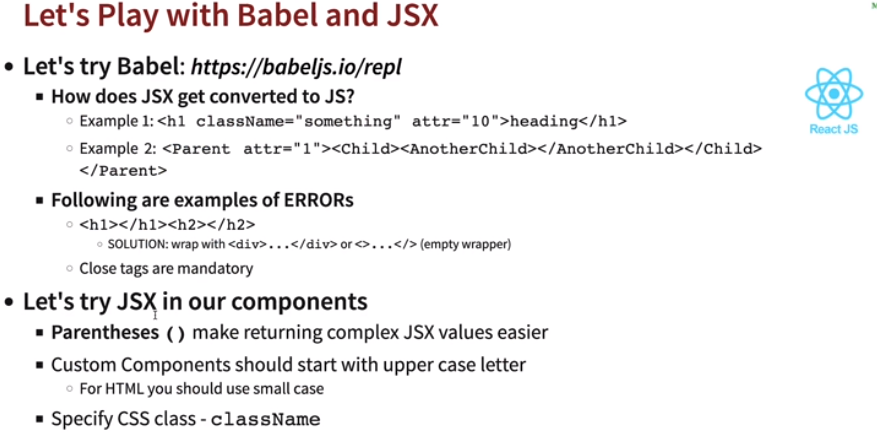
* + Only one Top level tag is allowed, otherwise if we have multiple tags in the component we need to use <></> or <div></div> to enclose.

** **

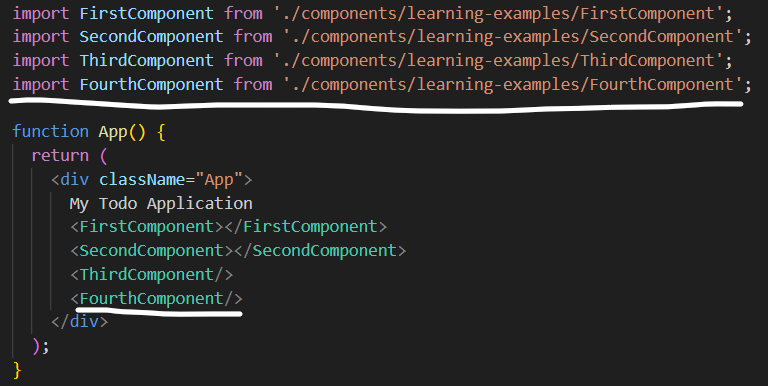
****

* **Babel:** Browsers now uses new JS (ES2022, ES2023….), to ensure compatibility with older versions of browsers, Babel is used. Babel is responsible for converting your JSX to JS.

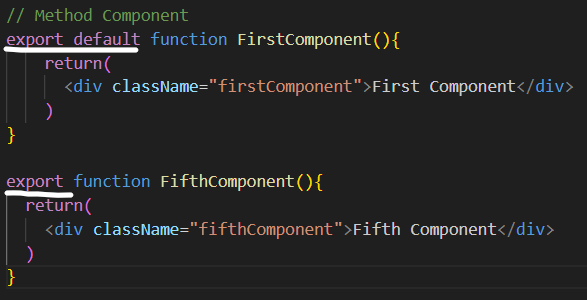
****

****

* **Exporting component from different files:** 
  + Import it in *App.js* as *import <Component> from ‘<Location>’;*

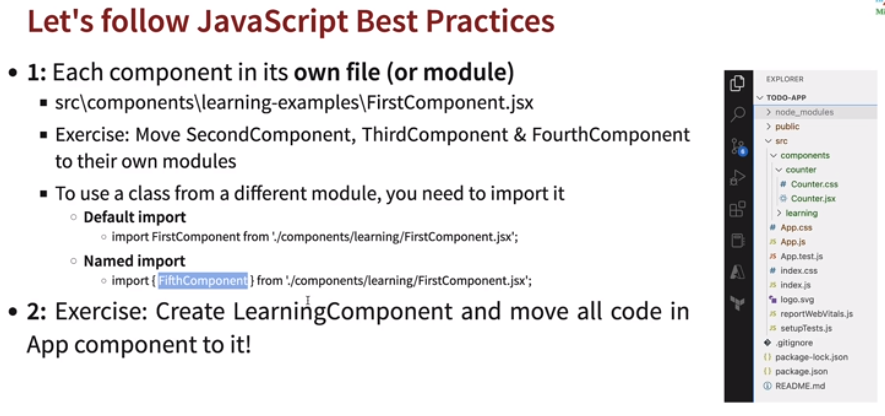
****

* + Use *export/export default* in the JSX file.

****

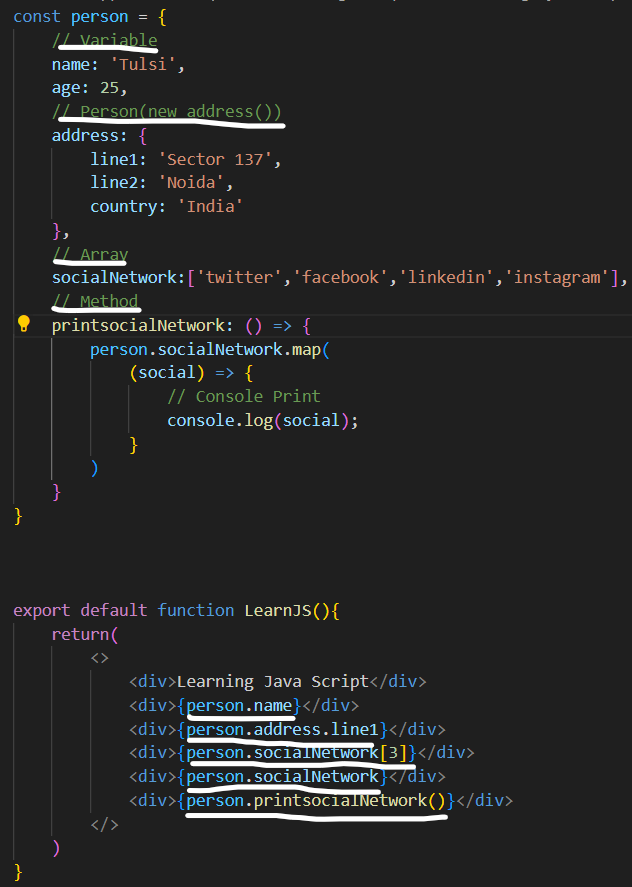
* + By default there’s only one *export default is allowed.* If we need to import another export component which isn’t an *export default,* while importing we need to use ***{}***and if ***{}*** is not used then only *export default will import.*

****

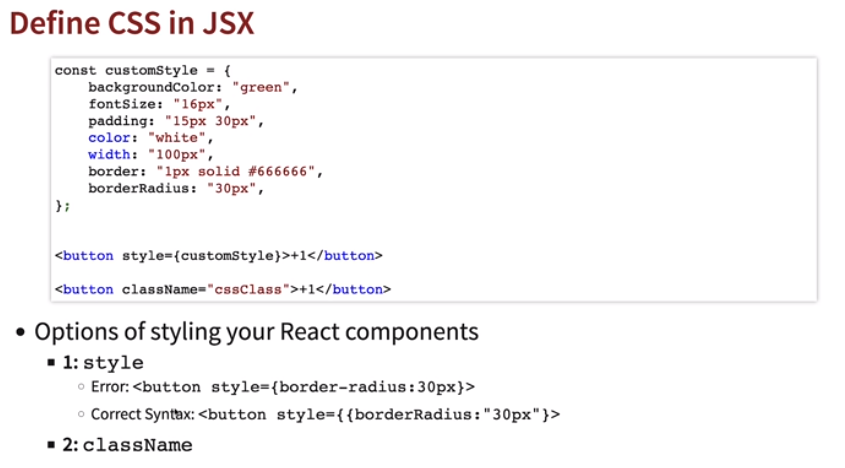
****

* **Learning JS:**

****

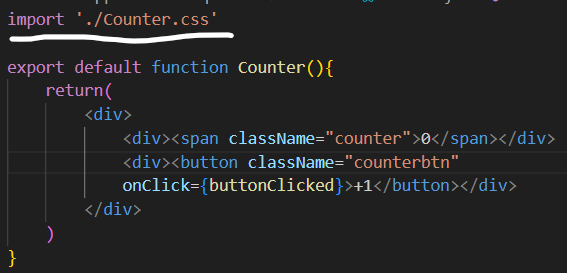
****

* **Styling (CSS):** In JSX if we define style inside JSX or element we need to use ***{{<style>}}***. Otherwise it will not work or other way is to store it in Java object and access as style variable.Also ***–*** are not allowed in CSS property so we need to use Capital first letter. Ex: *font-size: 🡪 fontSize:*

****

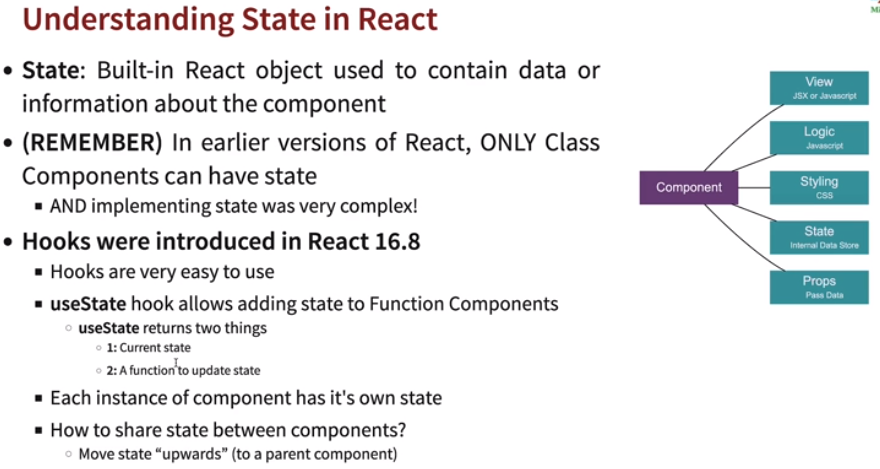
** **

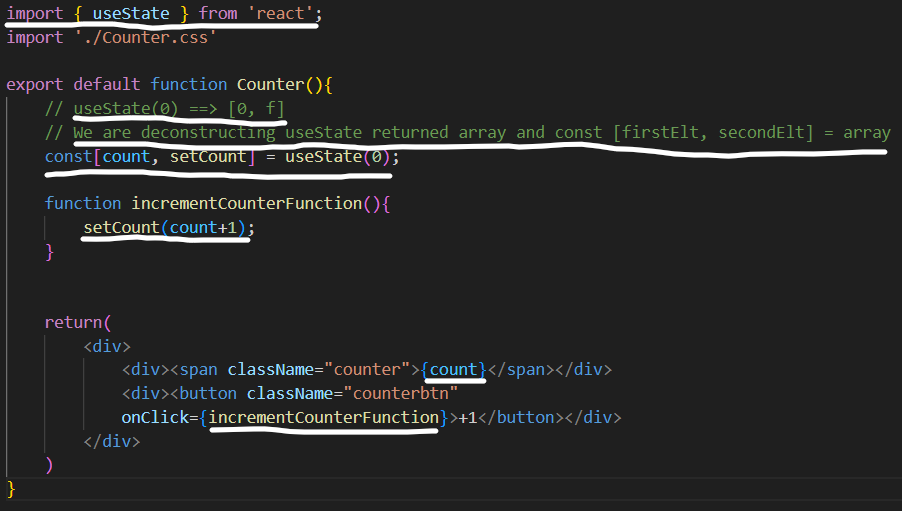
* + Importing from another CSS file, just need to add *import ‘<file>’.*

** **

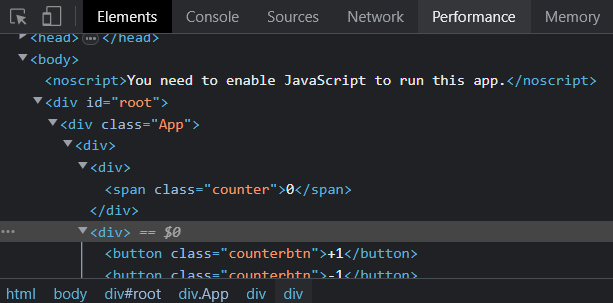
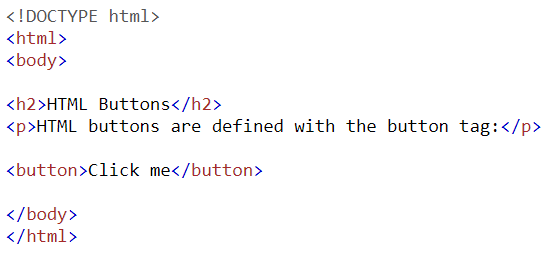
* **State:** Built in React objects to contain data or information of a component. The *useState* Hook can be used to keep track of *strings, numbers, booleans, arrays, objects, and any combination of these.* A *useState* will return or have array of 2 things, one is current state/value and other is function for changing its value.

****

****

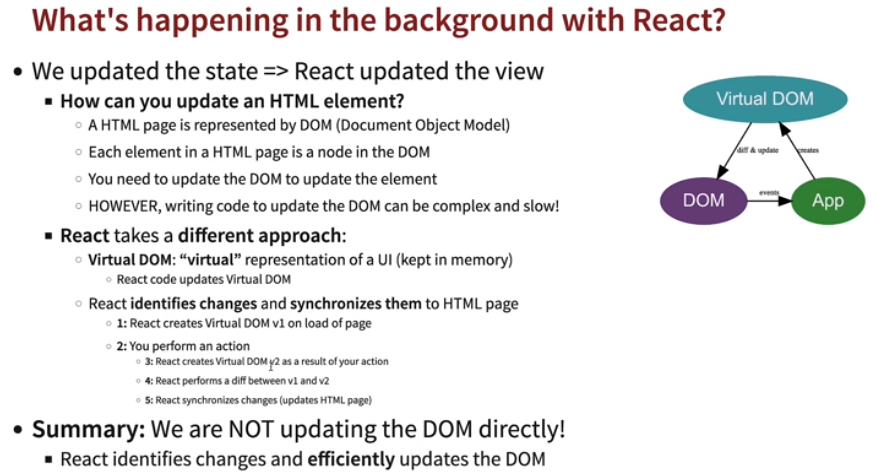
****

* **DOM (Document Object Model):** The Document Object Model (DOM) is a language-independent model made up of objects representing the structure of a document. HTML is one language for writing such documents.

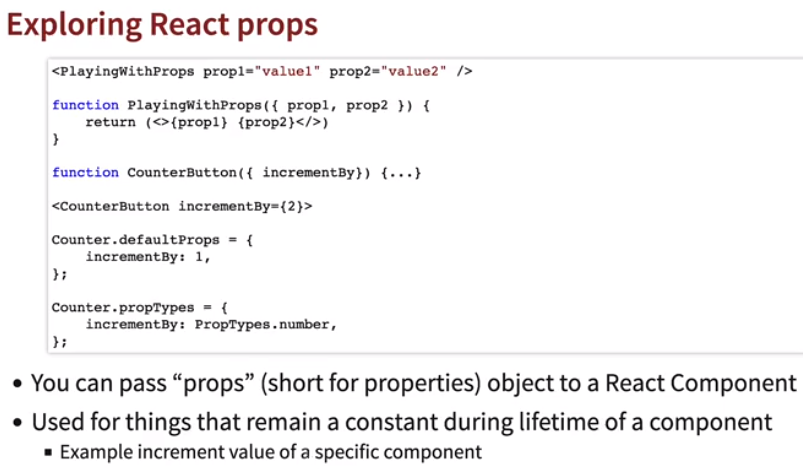
** **

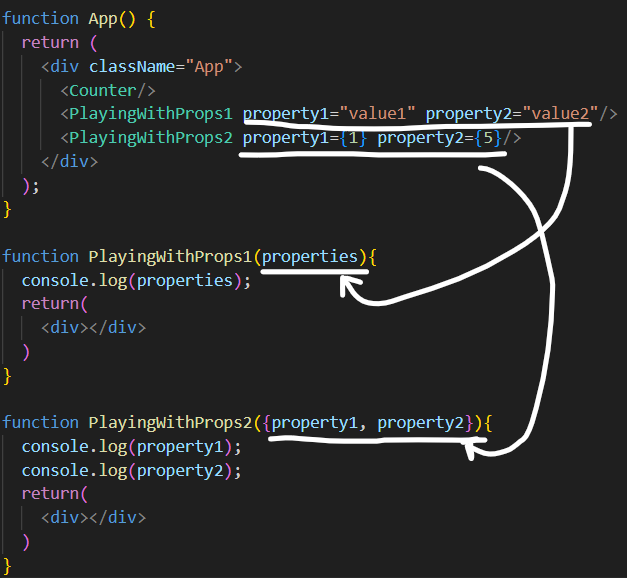
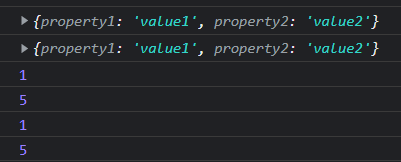
**DOM HTML**

* **How React updates HTML so fast or what’s happening in the background?**
  + In earlier approaches, we used to write the code to find the specific elements in HTML and write code to update its values.
  + React keeps the *virtual DOM in memory* when we make changes, React compares the changes b/w *v1(old vDOM)* and *v2(new vDOM)* and updates DOM with differences b/w *v1* and *v2*. React identifies changes and synchronizes with HTML pages. This makes React much more efficient.

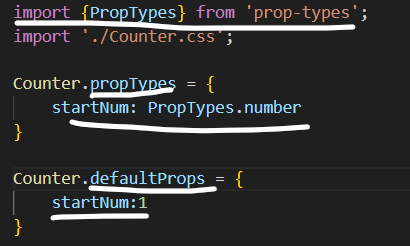
****

* **Props:** Props is short for properties are the values that remain constant and never change throughout. *<Component property1=”<val1>” property2=”<val2>”…../>.* Properties return an object of these values *(properties) 🡪 returns object of props*, or if we want to access values directly instead from object we need to have *{property1, property2…..}* defined in components.

****

** **

* Here we’re setting the properties attribute by specifying it’s default type should be number and default value should be *1*.
  + *Import {propTypes} from ‘prop-types’;*.
  + Then use as 🡪 *<Component>.propTypes = {<prop>: <attr>}.*

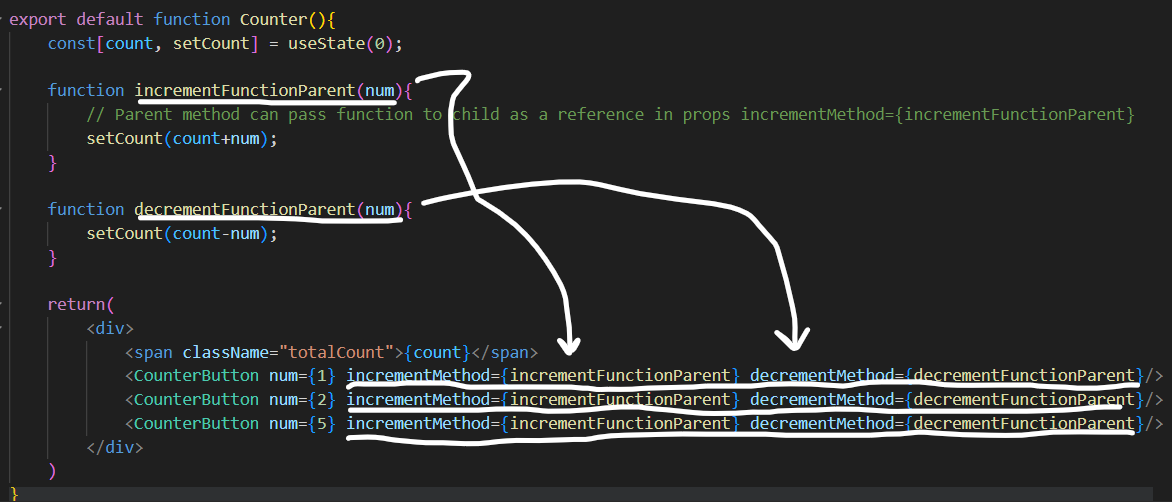
****

* In case we change the propType to some other let’s say String we’ll get error in console.

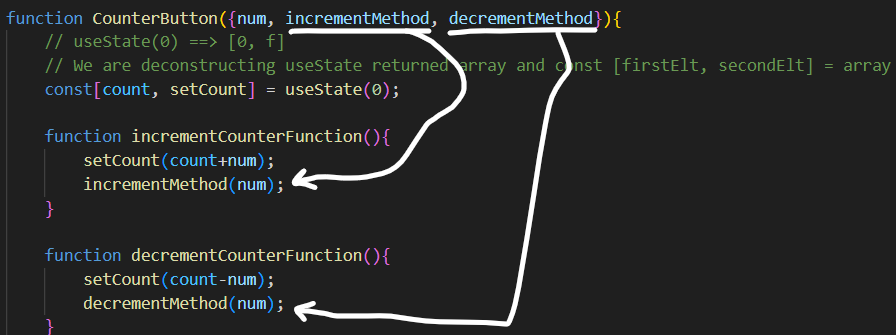
****

****

* **Passing Parent method to child:** Parent method can be passed as props into Child Method, or it can be directly called using *()=><ParentFunction(<arg>)>.*

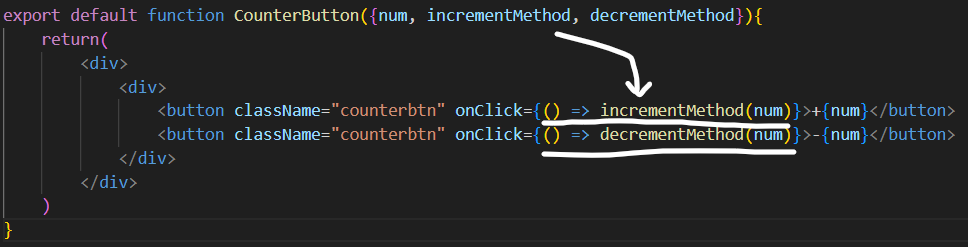
****

**Parent Class**

****

**Child Class**

**Or**

****

**Child Class**