Unit 6 –

**Front End Frameworks**

Digging deeper into some React concepts

State and Event Handling

1. **useState**: “hook” function that gives components state
   1. **returns a state variable and a setter function to update it**.
2. Setter functions should NOT be called into the main body of a component. This will lead to an infinite rendering loop.
   1. If the setter is hit within the main body of a component,
   2. It will update the state and force a re-render.
3. In our previous example:
   1. If “App” is a component which generates 3 “box” components
      1. The setter should **only** be in the “App” body
   2. But if we want to invoke it from children, we could put the setter inside a function and pass that function onto the children.
   3. ]
4. HTML Element Callbacks
   1. onClick
   2. onSubmit
   3. onChange
   4. etc –
   5. Remember – these are camel Cased in JSX.
5. Synthetic Events
   1. When the event callback is invoked, React creates a synthetic event and passes this ‘event” object as the first argument to the handler. From there, we can examine the event to discover things like the source element’s ID.
   2. Event.target
      1. This is provided to us every time an event occurs
      2. You can pass this into the setter function.
   3. If you want a specific div to disappear, you’d have to basically use this “event target” in order to get it to target specific things.
   4. And in React, rather than just passing the name of the function into, say, the functionality of a button, we can also pass in a parameter (for example eventObj) that we would then be able to use in the setter.
6. Each time there is a state change (a call to a setter) in a React app, a reconcilitation process is started.
   1. React will parse through the tree of the Virtual DOM and determine what is different, and patch it into the actual DOM.
   2. If each node in the list has a specified key, react should be able to find, insert, delete, and substitute changes with more ease.
7. Updating the DOM itself is not expensive, but painting the DOM visually to the screen is.
   1. React batches changes together to only “paint” once.
   2. When updating DOM, react only re-renders the sub tree.
   3. And further optimizations that can be set up.
8. Components should be **pure functions**
   1. Predictable returns the same JSX element when called with the same input;
   2. They should **never** change other code outside of their scope.
   3. But sometimes, side effects are necessary. And we will use something called “useEffect” to handle catches in a fetch.
   4. “useEffect” is designed to run side effects of rendering.
   5. Most commonly used to handle requests to external systems.
   6. And callbacks passed into UseEffect will run after the component has finished rendering.
   7. And you can customize useEffect to run its callback only if specifi ststae variables or props have updated via a “dependency list”