KEEPER APP PROJECT PART 2

TASK://Challenge. Render all the notes inside notes.js as a seperate Note component.

1. In App.js, pass some custom props :title and content Then pass the values that is in h1 and p

```
<Note title="This is the note title"
      content="his is the note content"
```

2. In Note.js, Insert the title and content in ht Note.js using props:

```
<h1>{props.title}</h1>
 <p>{props.content}</p>
```

THIS RENDERS THE APP THE SAME BUT How to I render multiple notes that is in the note.js: Use

Map function instead of creating multiple Note component

3. In notes.js export notes.js, In App.js, import notes.js

```
notes.js
31  ];
32
33  export default notes;
```

```
App.js
5   import notes from "../notes";
```

4. In, App.js, REPLACE <Note /> and map function {notes.map(createNotes)}

5. Create createNote function, it will get single noteItem parameter and return Note components
6. <Note /> component will have props title and content They will get the values using this noteItem. Make sure to use key prop to identify unique component:

```
function createNotes(noteItem){
  return <Note
  key={noteItem.key}
  title = {noteItem.title}
  content = {noteItem.content}/>
}
```

```
function App() {
  return (
    <div>
      <Header />
      <Note title="This is the note title"
      content="his is the note content"
      />
      <Footer />
    </div>
  );
}
```

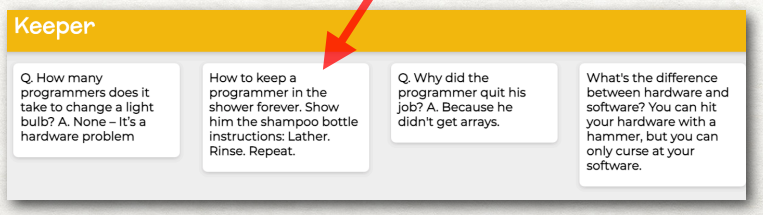
1

```
import React from "react";

function Note(props) {
  return (
    <div className="note">
      <h1>{props.title}</h1>
      <p>{props.content}</p>
    </div>
  );
}

export default Note;
```

2

```
App.js | notes.js | Note.js
src > components > App.js > App
1   import React from "react";
2   import Header from "./Header";
3   import Footer from "./Footer";
4   import Note from "./Note";
5   import notes from "../notes";
6
7   function createNotes(noteItem){
8     return <Note
9     key={noteItem.key}
10    title = {noteItem.title}
11    content = {noteItem.content}/>
12  }
13
14  function App() {
15    return (
16      <div>
17        <Header />
18        {/* <Note title="This is the note title"
19        content="his is the note content"
20        /> */}
21        {notes.map(createNotes)}
22        <Footer />
23      </div>
24    );
25  }
26
27  export default App;
```

```
notes.js
src > JS notes.js > ...
1   const notes =
2     {
3       key: 1,
```



KEEPER APP PROJECT PART 2

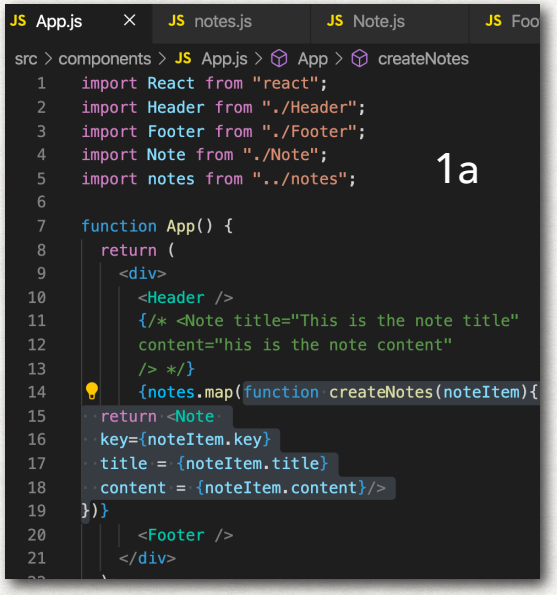
We can write this code in different ways:

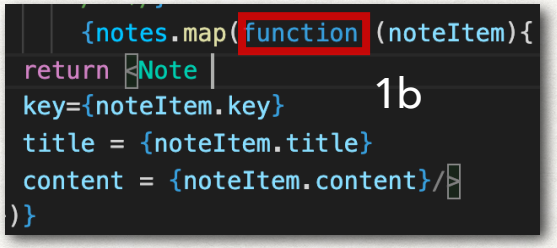
1a. We can put createNote function directly in the map function: So cut createNotes function and paste inside the map:

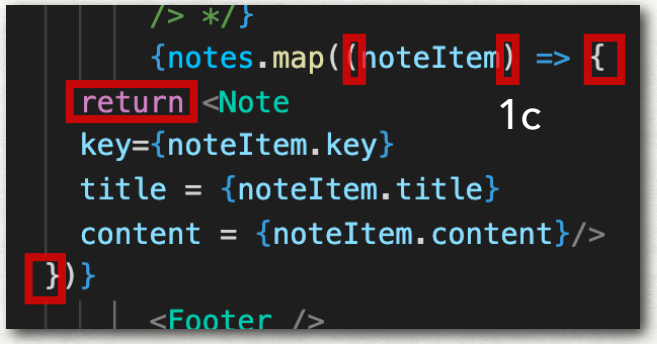
1b. I can make it anonymous ruction by deleting name:Delete createNotes

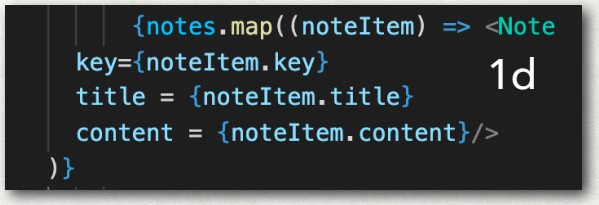
1c. Or Also remember we can use arrow functions, so we can simplify arrow to make it looks shorter: Delete function keyword, use =>

1d. Since we return only one single item, we can delete return, (), {}









14. CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

Think about log in screen. We will create a log in flow

When user is not log in, they should see the log in form, otherwise see the login version of the web

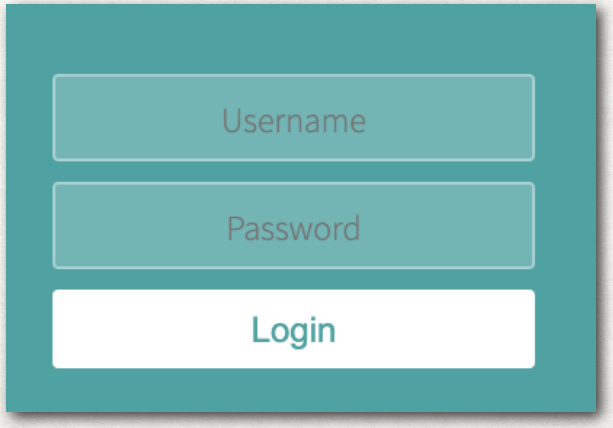
Import **nine-conditional-rendering**

npm install react-scripts start

npm install

npm start

TASK: We will render Hello only is user is logged on, otherwise show login form



CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

1. In to App.js Create a var isLoggedIn=true; This will be used to check if we loved in or not We will render Hello is isLoggedIn true, else render the form

2. In App.jsx, Create a function renderConditionally(). If user logs in, This will return <h1>Hello</h1>,otherwise this will return the log in form elements

3. Inside the App( ) function call renderConditionall function as js object:

4. Test your code: change isLogIn false and true, refresh page, and see it it works

```
import React from "react";

var isLoggedIn=false;
function renderConditionally(){
  if(isLoggedIn===true){
    return <h1>Hello</h1>;
    }else{
      return (
        <form className="form">
          <input type="text" placeholder="Username" />
          <input type="password" placeholder="Password" />
          <button type="submit">Login</button>
        </form>
      );}}

function App() {
  return  <div className="container">{renderConditionally()} </div>
}
export default App;
```

This is one way of doing that rendering conditionally

But Each component should have single responsibility

So we will create Login component for the entire form component

CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

Create Login.jsx and and add the form from App.jsx

And render <Login/> component in the renderConditionally function

App.jsx | Login.jsx

src > components > Login.jsx > default

```jsx
import React from "react";

function Login(){
    return   <form className="form">
    <input type="text" placeholder="Username" />
    <input type="password" placeholder="Password" />
    <button type="submit">Login</button>
  </form>
}

export default Login;
```

JS App.jsx | JS Login.jsx

src > components > JS App.jsx > renderConditio

```jsx
import React from "react";
import Login from './Login';

var isLoggedIn=false;
function renderConditionally(){
  if(isLoggedIn===true){
    return <h1>Hello</h1>;
    }else{
      return (
        <Login />
      );}}
```

CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

NOW EXTRACT THE INPUT COMPONENT

1. Create **Input.jsx component,** import react, Create Input function, return the input elements

2. In Login.jsx, import Input, and return <Input />

3. We need two props, one of username, one for password in <Input />

```
function Login(){
    return <form className="form">
   <Input type = "text"
      placeholder="Username"/>
     <Input type = "password"
      placeholder="Password"/>
    <button type="submit">Login</b
```

4. In Input.jsx To get eta values of the username and password properties in the Input components, We need to use props:

```
function Input(props){
    return <div>
            <input
            type={props.type}
            placeholder={props.placeholder} />
    </div>
```

5. Now the website looks good!!!

6. We CAN MAKE THIS CODE SHORTER??

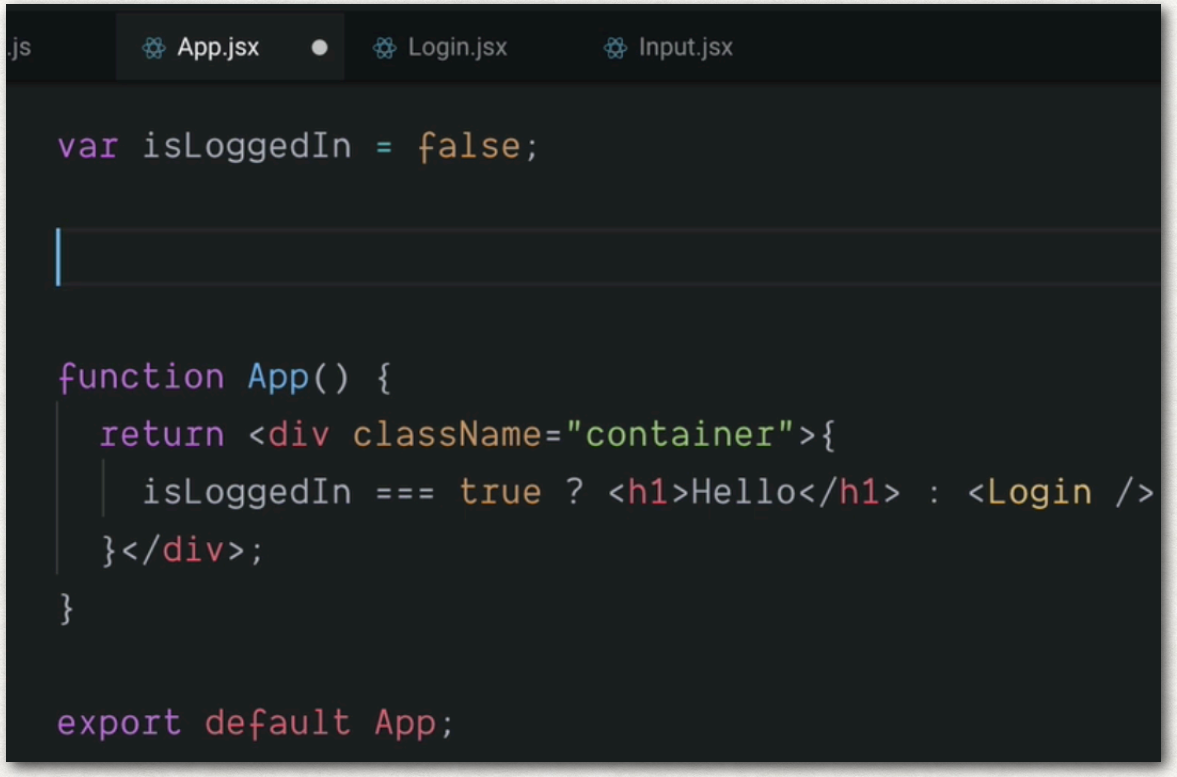
CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

Now instead of creating function and putting in the curly braces in App. Function, we will directly create the function inside

In jsx everything inside the curly braces must be an expression (ternary) NOT A STATEMENT(loops, switch, if else statement).

So inserting the renderConditionally function will not work. But we can use ternary

1. In App.jsx, Use ternary to render <h1>Hello</h1> if true, <Login/> if false. We can delete renderConditionally function and object now.



CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

Let's render current time is currentTime > 12, otherwise DON'T SHOW ANYTHING

```
function App() {
  return <div className="container">{
    // isLoggedIn ? <h1>Hello</h1> : <Login />
   currentTime > 12 ? <h1>Why are you still working?</h1> : null

  }</div>;
```

With and operator && What we are saying is if left side is true, render the right side, else don't render anything

App.jsx Login.jsx Input.jsx

```
var isLoggedIn = false;

const currentTime = new Date(2019, 11, 1, 18).getHours();

console.log(currentTime);


function App() {
  return <div className="container">{
    // isLoggedIn ? <h1>Hello</h1> : <Login />
   currentTime > 12 ? <h1>Why are you still working?</h1> : null

  }</div>;
}

export default App;
```

WORKS FINE. THERE IS && OPERATON IN JS WE CAN USE TO MAKE THE CODE EVEN SHORTER

CONDITIONAL RENDERING WITH TERNARY OPERATOR @ AND OPERATOR

With and operator && What we are saying is if left side is true, render the right side, else don't render anything

```
(EXPRESSION && EXPRESSION)
```

IF BOTH TRUE THEN RETURN TRUE ELSE RETURN FALSE

THIS MEANS IF 1st EXPRESSION IS FALSE, THEN THIS RETURNS FALSE

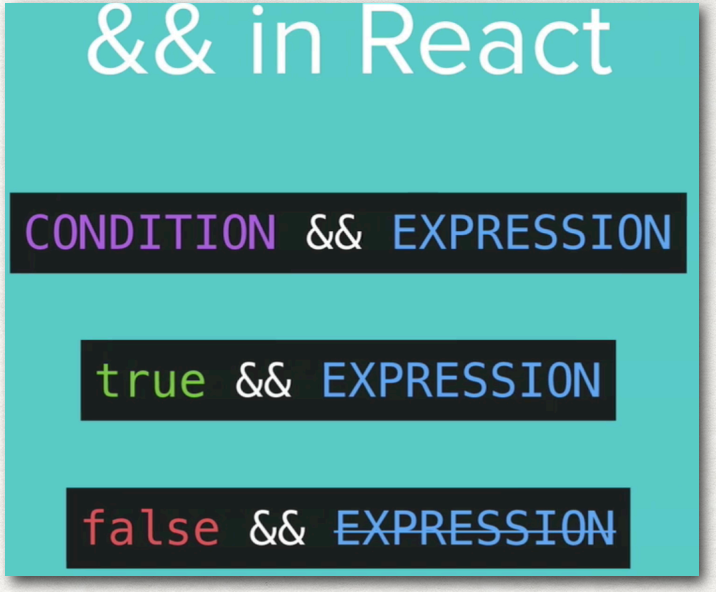
1. We can DELETE NULL and use && instead of ? operator

```
currentTime > 12 && <h1>Why are you still working?</h1>
```

This makes sense because if currentTime>12 left won't be checked

IN CONCLUSION, AS DEVELOPER, WE HAVE CHOICES AND WRITE DIFFERENT CODES THAT DOES SAME THINGS

DONE!!!!!!



App.jsx | Input.jsx | Login.jsx | index.html

src > components > App.jsx > App

```jsx
import React from "react";
import Login from "./Login";

var isLoggedIn=false;
const currentTime = new Date(2020,11,12,15).getHours();
console.log(currentTime);

function App() {
  return  <div className="container">
        {/* {isLoggedIn===true ? <h1>Hello</h1> : <Login />} */}
        {/* we can remove ===true */}
        {/* {isLoggedIn ? <h1>Hello</h1> : <Login />} */}
        {/* let's render a conditional message if time is greater tha
        { currentTime > 12 ? <h1>It is too late</h1> : null}
        {/* Below and operator does the same thing. In short but adva
        { currentTime > 12 && <h1>It is too late</h1>}
      </div>
}
export default App;
```

CONDITIONAL RENDERING HOMEWORK

```
//Challenge: Without moving the userIsRegistered variable,
//1. Show Login as the button text if userIsRegistered is true.
//Show Register as the button text if userIsRegistered is false.
//2. Only show the Confirm Password input if userIsRegistered is false.
//Don't show it if userIsRegistered is true.
```

Import conditional-rendering-practice

npm install react-scripts start

npm install

npm start

If user registered , show login screen

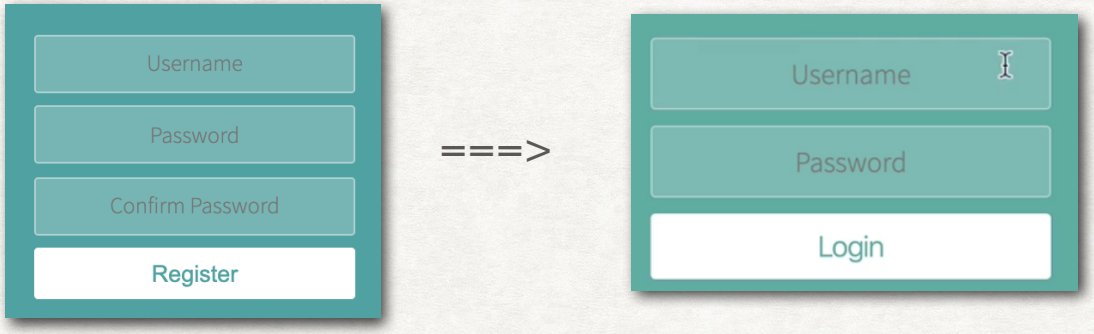
If not registered, show register screen

If on the register screen , show confirm password screen

If on the log in screen, do not show the confirm password screen

In Appjsx, when userIsRegistered = false we see username, password, confirm password

Change userIsRegistered=true, then you SHOULD SEE username, password, LogIn



CONDITIONAL RENDERING PRACTICE

//1. Show Login as the button text if userIsRegistered is true. Now it always shows Register button. Change the text "Register" that is inside App.jsx dependent of the value of userIsRegistered true or false. We will use props.

1. In App.jsx in add the props in the Form **<Form isRegistered={userIsRegistered}/>**

2. In Form.jsx, create props, use that props do conditional rendering of Register Text :

<button type="submit">{props.isRegistered ? "Login" : "Register"}</button>

NOW UI SHOULD RENDER LOGIN OR REGISTER BASED ON isRegister value

//2. Only show the Confirm Password input if userIsRegistered is false.//Don't show it if userIsRegistered is true.

Basically, when on Login screen, don't render confirm password input, When in the Register screen render confirm password

3. In Form.jsx add this where placeholder="Confirm Password is"

{props.isRegistered===false && (<input type="password" placeholder="Confirm Password" />) }

What this means is FIRST PART IS TRUE(props.isRegistered===false) is true render right side. DONT RENDER IT OTHERWISE

NOW THIS SHOULD WORK. CHANGE THE useIsRegistered true otr false to test

4. We can use ternary in the last part!.TEST IT.

```
{props.isRegistered === false ? (
  <input type="password" placeholder="Confirm Password" />
) : null}
```

5. FINALLY INSTEAD OF props.isRegistered===false CAN USE !props.isRegistered. TEST IT

DONE

App.jsx | Form.jsx | index.js

src > components > App.jsx > ...

```jsx
import React from "react";
import Form from "./Form";

var userIsRegistered = true;

function App() {
  return (
    <div className="container">
      <Form isRegistered={userIsRegistered}/>
    </div>
  );
}

export default App;
```

Form.jsx — conditional-rendering-practice

App.jsx | Form.jsx | index.js

src > components > Form.jsx > ...

```jsx
import React from "react";

function Form(props) {
  return (
    <form className="form">
      <input type="text" placeholder="Username" />
      <input type="password" placeholder="Password" />
      {/* {props.isRegistered===false && (<input type="password" placeho
      {/* BELOW IS SAME SLIGHTLY SHORTER */}
      {!props.isRegistered && (<input type="password" placeholder="Confi
      {/* BELOW IS HT ESECOND WAY */}
      {/* {props.isRegistered===false ? (<input type="password" placehol
      <button type="submit">{props.isRegistered ? "Login" : "Register"}<
    </form>
  );
}

export default Form;
```

REACT HOOKS -USESTATE

We will learn how to make our apps more interactive.

To do that we will learn the concept of STATE.

State is one of the fundamental concept in react

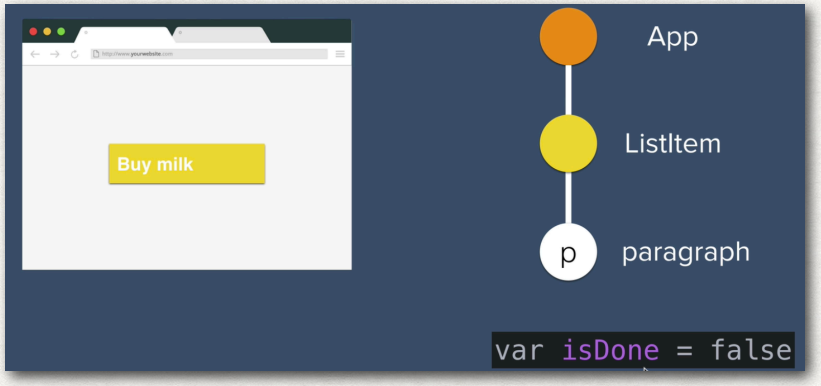
UI is actually a function of the state

When the state of a UI changes, we see different behavior of the app.

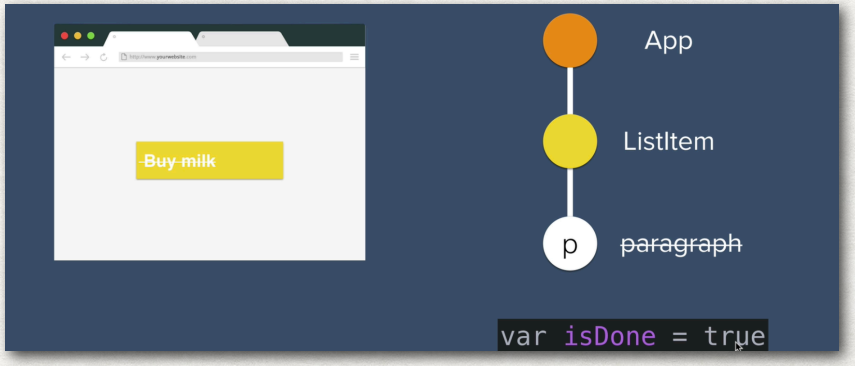
One of the way to change the state is, kept track of the condition of a variable

We will change the sate of the element depending on the isDone variables state.

In this practice we as we click on a button, we will increate the value on the number



USER CLICKS THEN CONDITION CHANGES



REACT HOOKS -USESTATE

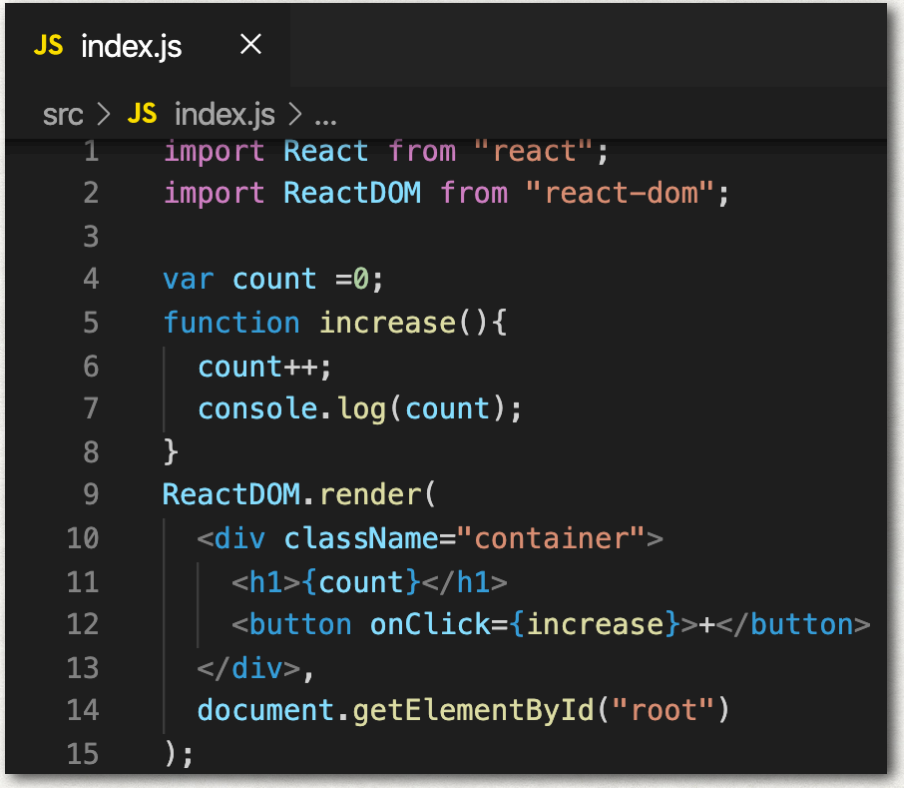
Import ten-usestate-hook,npm install react-scripts start,npm install,npm start

Now I got everything inside my index.js and rending a single div with h1 and button

Our goal is the button should trigger and update the number. We want count change so create a count variable

We want to trigger + button. In HTML, button on-click attribute to trigger click action BUT in JS it is onClick.

So use **onClick** attribute for the button and assign a function called increase. Increate function will increate the value of count by 1 when we clock on the + button. onClick will trigger an action and update the element



REACT HOOKS -USESTATE

In App.jsx.js,

1. Create App component and return the elements.

2.Create a variable count, initialize and use it in the h1 as js object

3.Use onClick attribute in the button and set it to a function called increase

4.Create increase function to increment the count

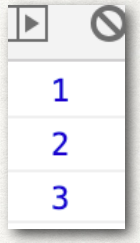
In index.js, call the App component

Now as we click the button, we should see the count is increasing on the console, But it is not rendering on the UI.

index.js | App.jsx

components > App.jsx > ...

```
1   import React from "react";
2
3   let count=0;
4   function increase(){
5     count++;
6     console.log(count);
7   }
8
9   function App() {
0     return  <div className="container">
1     <h1>{count}</h1>
2     <button onClick={increase}>+</button>
3   </div>;
4   }
5
6   export default App;
```



index.js | App.jsx

src > index.js

```
1   import React from "react";
2   import ReactDOM from "react-dom";
3   import App from './components/App'
4
5   ReactDOM.render(
6     <App />,
7      document.getElementById("root")
8   );
```

REACH HOOKS-USESTATE

1. In App.jsx, in the App function, Instead of var count = 0; we use useState hook

```
const state = useState(123);// 123 is starting state
console.log(state[0]);//return 123 on the console
```

2. import React, { useState } from "react";

3. To display on the <h1 element: <h1>{state[0]}</h1>. This me
   Whenever useState updated the ui automatically be updated

4. This is has coded. We will Destructure arrays and objects

5. To destrucre useState, let's map useState initial value to an array value:Below will render the initial value:

```
const [count] = useState(123);//assign 123 for count initial
<h1>{count}</h1>//Render count in the h1.UI SHOULD RENDER SAME
```

6. HOW DO WE INCREASE COUNT AS WE CLICK ON COUNT BUTTON???

7. useState(initialValue,function)

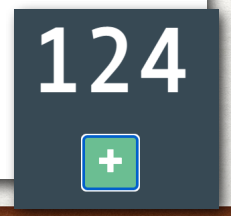
```
const [count,setCount] = useState(123);
function increase(){
  setCount(count+1)//increate the initial value one by one
}
<button onClick ={increase}>+</button>
```

6.

dex.js | JS App.jsx

components > JS App.jsx > ...

```
import React, { useState } from "react";

// let count=0;
// function increase(){
//   count++;
//   console.log(count);
// }

function App() {
  const [count,setCount] = useState(123);
  function increase(){
    setCount(count+1)
  }
  return  <div className="container">
  <h1>{count}</h1>
  <button onClick={increase}>+</button>
</div>;
}

export default App;
```



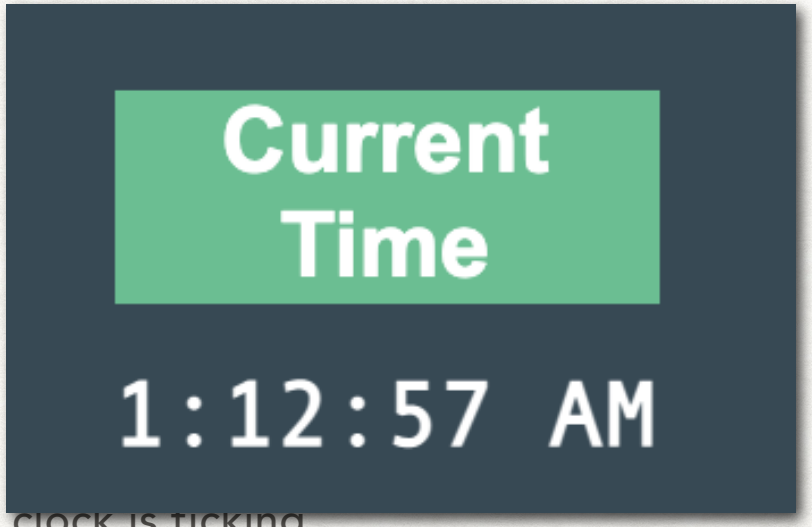
REACH HOOKS-USESTATE PRACTICE

YOUR TURN

Add a - sign and decrease the function

REACH HOOKS-USESTATE PRACTICE

RENDER THE CURRENT local TIME ON THE UI



Steps :

1. Create a button with text: Current Time
2. Style the button to match the style
3. Create an h1 and to get the current time.
4. Make sure to use create needed objects and the clock is ticking

NOTE : USE setInterval(function) TO SET RENDER THE UPDATE TIMES

REACT HOOKS-USESTATE PRACTICE

1. Create a button with text: Current Time

1. Style the button to match the style

```
<button style = {
  {fontSize:"30px",marginTop:"20%",width:"60%"}
  }>Current Time</button>
```

2. Create an h1 and to get the current time.

3.

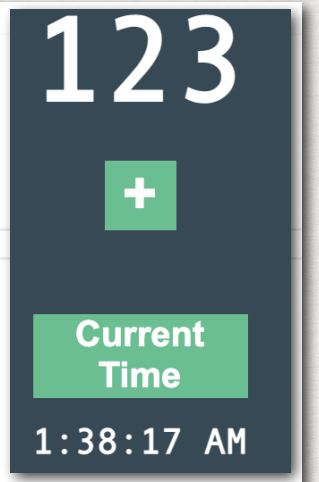
```
const now = new Date().toLocaleTimeString();
console.log(now);
<h1 style = {{fontSize:"30px"}}>{now}</h1>
```

```
//OR
const [time,setTime] = useState(new Date().toLocaleTimeString());
console.log(time);
```

```
//Creating a setInterval function that will update setTime every 1000 milisecond
//setInterval(setTime, 1000);//=> State Hook functions doesnt work in setInterval.
//That is why, we need a new function
setInterval(updateTime, 1000);
function updateTime(){
  const newTime = new Date().toLocaleTimeString();
  setTime(newTime);
}
<h1 style = {{fontSize:"30px"}}>{time}</h1>
```

components > JS App.jsx > ...

```
import React, { useState } from "react";

// let count=0;
// function increase(){
//   count++;
//   console.log(count);
// }
function App() {
  const [count,setCount] = useState(123);
  function increase(){
    setCount(count+1)
  }
  // const now = new Date().toLocaleTimeString();
  // console.log(now);
  // const [time,setTime] = useState(now);
  //OR
  const [time,setTime] = useState(new Date().toLocaleTimeString());
  console.log(time);
  //Creating a setInterval function that will update setTime every 1000 milisecond
  //setInterval(setTime, 1000);//=> State Hook functions doesnt work in setInterval.
  //That is why, we need a new function
  setInterval(updateTime, 1000);
  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);}

  return  <div className="container">
  <h1>{count}</h1>
  <button onClick={increase}>+</button>
  <br />
  <button style = {
    {fontSize:"30px",marginTop:"20%",width:"60%"}
    }>Current Time</button>
  <h1 style = {{fontSize:"30px"}}>{time}</h1>
</div>;}

export default App;
```



USESTATE HOOK PRACTICE

Import eleven-usestate-hook-practice

npm install react-scripts start, npm install, npm start

Or goal to to render the clock on the page and see the clock is running

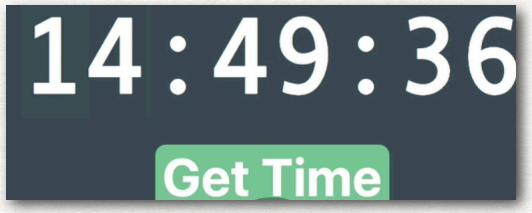
We break down the steps to learn the steps

TASK:

Part 1 : when we click on Get Time we should see the latest updated time

Part 2: we should always the updated time like a clock ticking

NOTE : USE setInterval(function) TO SET RENDER THE UPDATE TIMES




```
//Challenge:
//1. Given that you can get the current time
using:
let time = new Date().toLocaleTimeString();
console.log(time);
//Show the latest time in the <h1> when the Get
Time button
//is pressed.

//2. Given that you can get code to be called
every second
//using the setInterval method.
//Can you get the time in your <h1> to update
every second?

//e.g. uncomment the code below to see Hey
printed every second.
// function sayHi() {
//   console.log("Hey");
// }
// setInterval(sayHi, 1000);
```

USESTATE PRACTICE

Everything will happen App.jsx

So let's get the data inside the App function:

1. In App.jsx, creates constt now to get het current time

```
const now = new Date().toLocaleTimeString();
console.log(now);
```

1. In App.jsx, create a new function to destructure useState array that will hold the initial value of the time and a function that will update the time

```
  const [time, setTime]=useState(now);//starting value now. time=now
  console.log({time});
```

3. Use time, which has the update time in the <h1> to render it:

```
<h1>{time}</h1>
```

4. We will update the time when button clicked Use onClick and creat a function to update:

```
<button onClick={updateTime}>Get Time</button>
```

5. Create the updateTime function:

```
  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);
  }
```

5. Now whenever I click on the on Get Time button, I will get he new time. PART1 IS DONE. TEST

6. We will use setInterval function to display int he first line of the App function

```
  setInterval(updateTime, 1000);//MEANS update updateTime function every 1000 millisecond
```

7. It will create a new constant set to the current time and update our time variable that is used in h1 every single second! So we will see dynamic refreshing time

```
import React, { useState } from "react";

function App() {

  setInterval(updateTime, 1000);

  const now = new Date().toLocaleTimeString();
  // console.log(now);

  const [time, setTime] = useState(now);
  // console.log({time});

  function updateTime(){
    const newTime = new Date().toLocaleTimeString();
    setTime(newTime);
  }

  return (
    <div className="container">
      {/* <h1>TIME</h1> */}
      <h1>{time}</h1>
      <button onClick={updateTime}>Get Time</button>
    </div>
  );
}

export default App;
```