**Challenge 1: DISPLAY JSX**

function App() {

**return (**

**<h1>Hello<h1/>**

**);**

}

ReactDOM.render(<App />, document.getElementById("root"));

**Challenege 2: DISPLAY ARRAY OF USERS USING MAP FUNCTION**

const users = [

{ name: "John Doe", id: 1 },

{ name: "Jane Doe", id: 2 },

{ name: "Billy Doe", id: 3 }

];

**const renderedUsers = users.map((user) =>**

**<div key={user.id}>**

**{user.name}**

**</div>**

**);**

function App() {

return (

<>

<h3>User names</h3>

<ul**>{renderedUsers}</**ul>

</>

);

}

ReactDOM.render(<App />, document.getElementById("root"));

**CHALLENGE 3: SHOW/HIDE (TOGGLE) TEXT USING useState HOOK**

import React from 'react';

import {useState} from 'react';

export function App() {

  const [show, setShow] = useState(true);

**const handleOnClick = () => {**

**setShow(!show);**

**}**

  return (

    <>

      <button ***onClick*={handleOnClick**}>Hide Element Below</button>

**{show &&** <div>Toggle Challenge</div> }

    </>

  );

}

**CHALLENGE 4: User should be able to type in any characters on input and those character should show in the browser.**

import React from 'react';

import {useState} from 'react';

export function App() {

  const [text, setText] = useState("");

  //const handleOnSubmit = (*event*) => {

  // event.preventDefault();

  //  setText(event.target.value); }

**const handleOnChange = (*event*) => {**

**event.preventDefault();**

**setText(event.target.value);**

**};**

  return (

    <>

      {*/\* <form onSubmit={handleOnSubmit}>*

*<input type="text" placeholder="Enter Text" />*

*<p>{text}</p>*

*</form> \*/*}

      <input ***onChange*={handleOnChange}** *type*='text' *placeholder*='enter text' ***value*={text}** />

      <p**>{text}**</p>

    </>

  );

}

**CHALLENGE 5: DISABLE BUTTON TILL USER ENTERS ATLEAST A SINGLE CHARACTER IN INPUT.**

import React from 'react';

import {useState} from 'react';

export function App() {

  const [text, setText] = React.useState("");

**const handleOnChange = (*e*) => {**

**setText(e.target.value);**

  };

  return (

    <>

      <h3>Disable Button Challenge</h3>

      <input ***onChange*={handleOnChange}** *type*="text" *value*={text}/>

      <button ***disabled*={text.length < 1}**>Submit</button>

    </>

  );

}

**CHALLENGE 6: UPDATE PARENT BY CHILD**

function Child(**{setValue})** {

**const handleOnClick = () => {**

**setValue('Updated by child')**

**};**

return (

<>

<div>Child</div>

<button **onClick={handleOnClick}>**Change Parent Value</button>

</>

);z

}

function Parent() {

const [value, setValue] = React.useState(

"I need to be updated from my child"

);

return (

<>

<h3>Update Parent State Challenge (Using Callback)</h3>

<div className="wrapper">

<div>Parent</div>

<div className="box-wrapper">{value}</div>

</div>

<div className="wrapper">

<Child **setValue={setValue}/>**

</div>

</>

);

}

ReactDOM.render(<Parent />, document.getElementById("root"));

**CHALLENGE 7: DISPLAY CHILD CONTENT IN PARENT COMPONENT**

function Child() {

return <div>This is children content</div>;

}

// Add code only here

function Parent() {

return (

<div>

<h3>Parent Component</h3>

**<Child />**

</div>

);

}

function App() {

return (

<Parent>

<Child />

</Parent>

);

}

ReactDOM.render(<App />, document.getElementById("root"));

**CHALLENGE 8: SUM OF TWO INPUTS**

function App() {

const [number1, setNumber1] = React.useState();

const [number2, setNumber2] = React.useState();

const [total, setTotal] = React.useState(0);

**const handleOnChangeNum1 = (e) => {**

**setNumber1(+e.target.value);**

**//we use + to convert string to a number**

**};**

**const handleOnChangeNum2 = (e) => {**

**setNumber2(+e.target.value);**

**//we use + to convert string to a number**

**};**

**const handleOnClick = () => {**

**setTotal(number1 + number2);**

**}**

return (

<div>

<h2>Adding Two Numbers</h2>

<input **onChange = {handleOnChangeNum1}** placeholder="First Number" type="number" **value={number1}** />

<input **onChange = {handleOnChangeNum2}** placeholder="Second Number" type="number" **value={number2}** />

<button **onClick={handleOnClick**}>Add Two Numbers</button>

<p>Total: **{total**}</p>

</div>

);

}

ReactDOM.render(<App />, document.getElementById("root"));

**CHALLENGE 9: Counter app using usestate**

const App = () => {

const [count, setCount] = React.useState(0);

**const handleIncrement = () => {**

**setCount(count+1);**

**};**

**const handleDecrement = () => {**

**setCount(count-1);**

};

return (

<div>

<h2>Counter: {count}</h2>

<button **onClick={handleIncrement}>**Increment</button>

<button **onClick={handleDecrement}>**Decrement</button>

</div>

);

};

ReactDOM.render(<App />, document.getElementById("root"));

**CHALLENGE 10: FETCH DATA FROM API**

function App() {

const [userData, setUserData] = React.useState({});

**const fetchData = async () => {**

**const response = await fetch(url);**

**const jsonData = await response.json();**

**setUserData(jsonData);**

**};**

**React.useEffect (() => {**

**fetchData();**

**},[]);**

// No need to touch code below

return (

<div className="App">

<h2>User Data</h2>

<p>

<strong>Name: </strong>{" "}

{userData.name || "(Need to populate name here)"}

</p>

<p>

<strong>Website: </strong>

{userData.website || "(Need to populate website here)"}

</p>

<p>

<strong>Email: </strong>

{userData.email || "(Need to populate email here)"}

</p>

<p>

<strong>Phone: </strong>

{userData.phone || "(Need to populate phone here)"}

</p>

</div>

);

}

ReactDOM.render(<App />, document.getElementById("root"));