HOOKS

Setup ::

Continue with the setup used with the previous React stages

STATE HOOK

Stage 1

```
<script src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>

<div id="root">
</div>

<script type="text/babel">
   const { useState } = React;

   function ClickCounter() {
       const [count, setCount] = useState(0);

       return (
           <div>
               <p>You clicked {count} times</p>
               <button onClick={() => setCount(count + 1)}>
                   Click me
               </button>
           </div>
       );
   }

   ReactDOM.render(
       <ClickCounter />,
       document.getElementById("root")
   );
</script>
```

Stage 2

```
<script src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>

<div id="root">
```

```
</div>

<script type="text/babel">
const { useState } = React;

function Stack(props)
{
   const [stk, setStk] = useState([]);
   const [i,setI] = useState("");

   const push = () => {
       let pi = parseInt(i);
       setI("");
       if(!Number.isNaN(pi))
       {
           stk.unshift(pi);
           setStk([...stk]);
       }
       else
       {
           console.log("Only Numbers allowed");
       }
   };

   const pop = () => {
       stk.shift();
       setStk([...stk]);
   };

   return (
       <div>
           <h2> { props.name? props.name : "Stack" } </h2>
           <div>
               {
                   stk.map( v =>(
                                   <div>
                                       <span>{v}</span>
                                       <br></br>
                                   </div>
                               )
                           )
               }
           </div>
           <input type="text" value={i} onChange={e =>
```

```
            <input type="text" value={i} onChange={e =>
setI(event.target.value)} />
            <button onClick={push}>
                push
            </button>
            <button onClick={pop}>
                pop
            </button>
        </div>
    );
}

ReactDOM.render(
    <Stack name="Stack 1"/>,
    document.getElementById("root")
);
</script>
```

Effect Hook

Stage 1

```
script src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
<div id="root">
</div>
<script type="text/babel">
  const { useState, useEffect } = React;
  function ClickCounter() {
      const [count, setCount] = useState(0);

      useEffect(() => {
          !count? console.log("Component Mounted!") :
                  console.log("Component Remounted!");
      });

      return (
          <div>
              <p>You clicked {count} times</p>
              <button onClick={() => setCount(count + 1)}>
                  Click me
              </button>
          </div>
      );
  }
  ReactDOM.render(
      <ClickCounter />,
      document.getElementById("root")
  );
</script>
```

Stage 2

```
<script src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
<div id="root">
</div>
<script type="text/babel">
  const { useState, useEffect } = React;
  function ClickCounter() {
      const [count, setCount] = useState(0);

      useEffect(() => {
          !count? console.log("Component Mounted!") :
                  console.log("Component Remounted!");
      });

      useEffect(() => {
          !count? console.log("Component Mounted 2!") :
                  console.log("Component Remounted 2!");
      });

      return (
          <div>
              <p>You clicked {count} times</p>
              <button onClick={() => setCount(count + 1)}>
                  Click me
              </button>
          </div>
      );
  }
  ReactDOM.render(
      <ClickCounter />,
      document.getElementById("root")
  );
</script>
```

Stage 3

```
script src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
<div id="root">
</div>
<script type="text/babel">
  const { useState, useEffect } = React;
  function ClickCounter() {
      const [count, setCount] = useState(0);

      useEffect(() => {
          !count? console.log("Component Mounted!") :
                  console.log("Component Remounted!");

           return () => {
               console.log("Component Unmounted!");
           }
      });

      return (
          <div>
              <p>You clicked {count} times</p>
              <button onClick={() => setCount(count + 1)}>
                  Click me
              </button>
          </div>
      );
  }
  ReactDOM.render(
      <ClickCounter />,
      document.getElementById("root")
  );
</script>
```

Reducer Hook

Stage 1

```
<script src="https://unpkg.com/react@16/umd/react.development.js"
```

```
ossorigin></script>
ript src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>

<div id="root">
</div>

<script type="text/babel">
   const { useReducer } = React;

   const initialState = { count: 0 };

   function reducer(state, action) {
       switch (action.type) {
           case 'increment':
               return { count: state.count + 1 };
           case 'decrement':
               return { count: state.count - 1 };
           case 'reset':
               return { count: 0 };
           default:
               throw new Error();
       }
   }

   function ClickCounter() {
       const [state, dispatch] = useReducer(reducer,
initialState);

       return (
           <div>
               <h3>{state.count}</h3>
               <button onClick={() => dispatch({ type:
"increment" })}>+</button>
               <button onClick={() => dispatch({ type:
"decrement" })}>-</button>
               <button onClick={() => dispatch({ type: "reset"
})}>Reset</button>
           </div>
       );
   }
```

```
ReactDOM.render(
    <ClickCounter />,
    document.getElementById("root")
);
</script>
```

Stage 2

```
<script src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>

<div id="root">
</div>

<script type="text/babel">
   const { useReducer,useState } = React;

   const initialState = { balance: 0, lt:0 };

   function reducer(state, action) {
       switch (action.type) {
           case 'deposit':
               return { balance: state.balance + action.value,
lt: action.value };
           case 'withdraw':
               return { balance: state.balance - action.value,
lt: -action.value };
           default:
               throw new Error();
       }
   }

   function Piggybank() {

       const [state, dispatch] = useReducer(reducer,
initialState);
       const [i,setI] = useState("");

       const handleAction = (type) => {
```

```
        const handleAction = (type) => {
            let v = parseInt(i);
            setI("");
            if(!Number.isNaN(v))
                dispatch({type: type,value: v});
            else
                console.log("Only Numbers allowed");
        };

        return (
            <div>
                <h3>Balance: {state.balance}</h3>
                <h3>Last Transaction: {state.lt}</h3>
                <input value={i} onChange=
{e=>setI(e.target.value)}></input>
                <button onClick={() =>
handleAction("deposit")}>Deposit</button>
                <button onClick={() =>
handleAction("withdraw")}>Withdraw</button>
            </div>
        );
    }

    ReactDOM.render(
        <Piggybank />,
        document.getElementById("root")
    );
</script>
```

Stage 3

```
<script src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-
dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
<script src="https://unpkg.com/@material-
ui/core@latest/umd/material-ui.development.js"></script>

<div id="root">
</div>

<script type="text/babel">
```

```
ript type="text/babel">
const { Button, TextField, Card, CardContent, Typography,
le, TableBody, TableCell, TableContainer, TableHead, TableRow,
Paper } = MaterialUI;
   const { useReducer, useState } = React;

   const initialState = { balance: 0, lt: 0 };

   function reducer(state, action) {
       switch (action.type) {
           case 'deposit':
               return { balance: state.balance + action.value,
lt: action.value };
           case 'withdraw':
               return { balance: state.balance - action.value,
lt: -action.value };
           default:
               throw new Error();
       }
   }

   function Piggybank() {

       const [state, dispatch] = useReducer(reducer,
initialState);
       const [i, setI] = useState("");
       const [e, setE] = useState(false);

       const handleAction = (type) => {

           let v = parseInt(i);
           setI("");
           if (!Number.isNaN(v)) {
               setE(false);
               dispatch({ type: type, value: v });
           }
           else {
               setE(true);
           }
       };

       return (
           <div>
               <Card>
```

```
                    <CardContent>
                        <Typography variant="h5" component="h2">
                            Piggybank
                        </Typography>
                        <br />
                        <TableContainer component={Paper}>
                            <Table>
                                <TableHead>
                                    <TableRow>

<TableCell>Details</TableCell>
                                        <TableCell
>Amount</TableCell>
                                    </TableRow>
                                </TableHead>
                                <TableBody>
                                    <TableRow>
                                        <TableCell component="th"
scope="row">
                                            Balance
                                        </TableCell>
                                        <TableCell >
{state.balance}</TableCell>
                                    </TableRow>
                                    <TableRow>
                                        <TableCell component="th"
scope="row">
                                            Last Transaction
                                        </TableCell>
                                        <TableCell >{state.lt}
</TableCell>
                                    </TableRow>
                                </TableBody>
                            </Table>
                        </TableContainer>
                        <br />
                        <br />
                        <Button onClick={() =>
handleAction("deposit")} variant="contained" color="primary">
Deposit </Button>
                         
                        <TextField error={e} value={i} onChange={e
=> setI(e.target.value)} size="small" label="value"
```

```
    set(e.target.value)} size="small" label="value
    riant="outlined" />
                         
                        <Button onClick={() =>
handleAction("withdraw")} variant="contained" color="secondary">
Withdraw </Button>
                    </CardContent>
                </Card>
            </div>
        );
    }

    ReactDOM.render(
        <Piggybank />,
        document.getElementById("root")
    );
</script>
```