import {useState} from 'react'

or

const state = React.useState();

* if we do not use Usestate we would have had to rerender the component again to see the changes.

* const [a, b, c] = [1, 2, 3]

now a = 1 , b = 2 , c = 3

* const [count, setCount] = useState(1);

value function

count = 1

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

function App() {
  let now = new Date().toLocaleTimeString();
  let [time,setTime] = useState(now);
  function update(){
    setTime(time = new Date().toLocaleTimeString());
  }
  setInterval(update,1000);
  return (
    <div className="container">
      <h1>{time}</h1>
      <button onClick={update}>Get Time</button>
    </div>
  );
}

export default App;
```

Destructuring : How useState works?

import animals from "./data"

* const [cat, dog] = animals;

cat = animals[0]
dog = animals[1]

→ These names have to match the names which are there in animals

* Here as Cat & dog in a Object

const {name, sound} = cat;

cat.name
cat.sound = 'meow'

or animals[0].sound = "meow"

const {name: catName, sound: catSound} = cat

* Here we changed the name to CatName and sound to CatSound and now we will call

cat.CatName
cat.catSound

* const {name = "Fluffy", sound = "Purr"}

Here when we console.log(name) we will get meow as it is stored in the animals but if it was not present we would get Fluffy

* const [cat, dog] = animals;

object inside a object

const {name, sound, feeding Requirements: {food, water}} = cat;

// Destructuring a destructured Object

```
function useAnimals(animal) {
  return [
    animal.name,
    function() {
      console.log(animal.sound);
    }
  ];
}

export default animals;
```

export {useAnimals};

* now

import {useAnimals} from './data';

import Animals from './data';

const [cat, dog] = animals

const [animal, makeSound] = useAnimals(cat)

console.log(animal) cat

console.log(makeSound) meow

* Challenge

```
const cars = [
  {
    model: "Honda Civic",
    coloursByPopularity: ["black", "silver"],
    speedStats: {
      topSpeed: 140,
      zeroToSixty: 8.5
    }
  },
  {
    model: "Tesla Model 3",
    coloursByPopularity: ["red", "white"],
    speedStats: {
      topSpeed: 150,
      zeroToSixty: 3.2
    }
  }
];

export default cars;
```

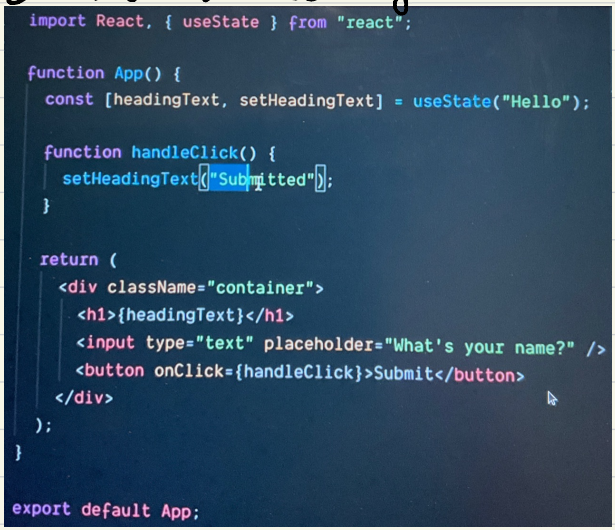
```
const [honda, tesla] = cars;

const {
  speedStats: { topSpeed: hondaTopSpeed }
} = honda;
const {
  speedStats: { topSpeed: teslaTopSpeed }
} = tesla;

const {coloursByPopularity: [hondaTopColour]} = honda;
const {coloursByPopularity: [teslaTopColour]} = tesla;
```

```
ReactDOM.render(
  <table>
    <tr>
      <th>Brand</th>
      <th>Top Speed</th>
      <th>Top Colour</th>
    </tr>
    <tr>
      <td>{tesla.model}</td>
      <td>{teslaTopSpeed}</td>
      <td>{tesla.topColour}</td>
    </tr>
    <tr>
      <td>{honda.model}</td>
      <td>{hondaTopSpeed}</td>
      <td>{honda.topColour}</td>
    </tr>
  </table>,
  document.getElementById("root")
);
```

Output

Brand	Top Speed	Top Colour
Tesla Model 3	150	red
Honda Civic	140	black

Event Handling



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

function App() {
  const [headingText, setHeadingText] = useState("Hello");

  function handleClick() {
    setHeadingText("Submitted");
  }

  return (
    <div className="container">
      <h1>{headingText}</h1>
      <input type="text" placeholder="What's your name?" />
      <button onClick={handleClick}>Submit</button>
    </div>
  );
}

export default App;
```

```jsx
const [mouseHover, setMouseHover] = useState(false);

function MouseOver() {
  setMouseHover(true);
}

function MouseOut() {
  setMouseHover(false);
}

<button onMouseOver={MouseOver} onMouseOut={MouseOut} style={{backgroundColor: mouseHover ? "black" : "white"}}>Submit</button>
```

OnChange event

↓

passes a Object

```
function App() {
  const [name, setName] = useState("");

  function handleChange(event) {
    console.log(event.target.value);
    setName(event.target.value);
  }

  return (
    <div className="container">
      <h1>Hello {name}</h1>
      <input
        onChange={handleChange}
        type="text"
        placeholder="What's your name?"
        value={name}
      />
      <button>Submit</button>
    </div>
  );
```

<button onClick={Submit}>

function Submit() {

setName(

}

→ to define that corresponding

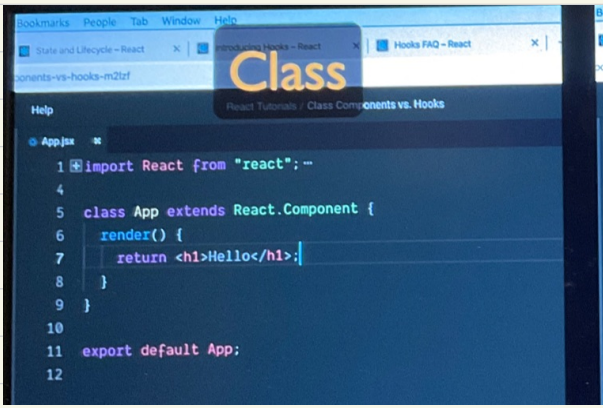
the name in h1 and value are to the same "name" (to match up)

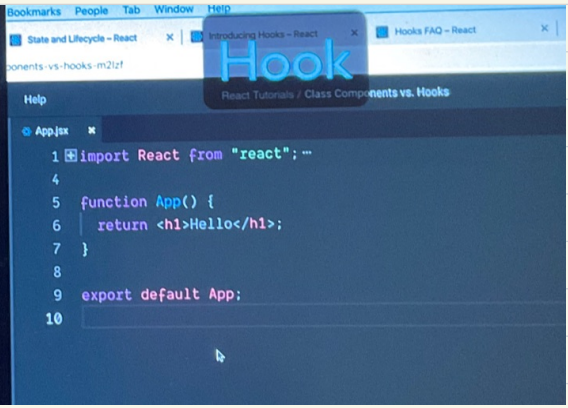
* never use event inside a setState

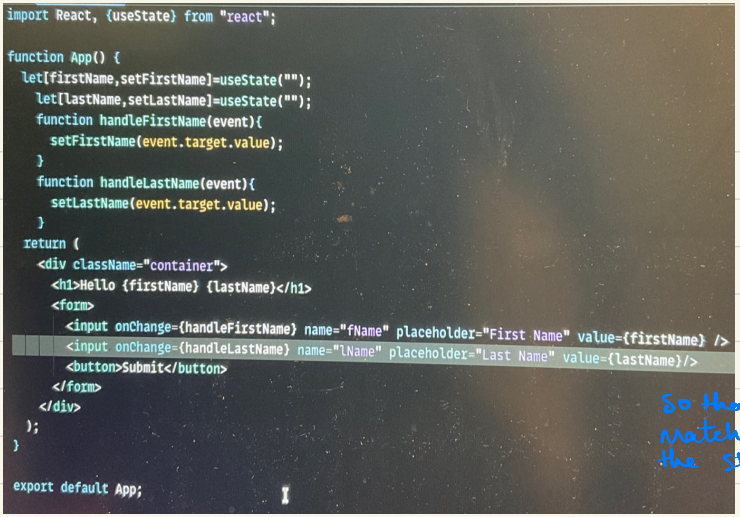
```jsx
import React, { useState } from "react";

function App() {
  const [name, setName] = useState("");
  const [headingText, setHeading] = useState("");

  function handleChange(event) {
    console.log(event.target.value);
    setName(event.target.value);
  }

  function handleClick(event) {
    setHeading(name);

    event.preventDefault();
  }

  return (
    <div className="container">
      <h1>Hello {headingText}</h1>
      <form onSubmit={handleClick}>
        <input
          onChange={handleChange}
          type="text"
          placeholder="What's your name?"
          value={name}
        />
        <button type="submit">Submit</button>
      </form>
    </div>
  );
}
```

→ will prevent default behaviour of forms being rendered after submission

* Default behaviour of <form></form> is they refresh on post requests or submit

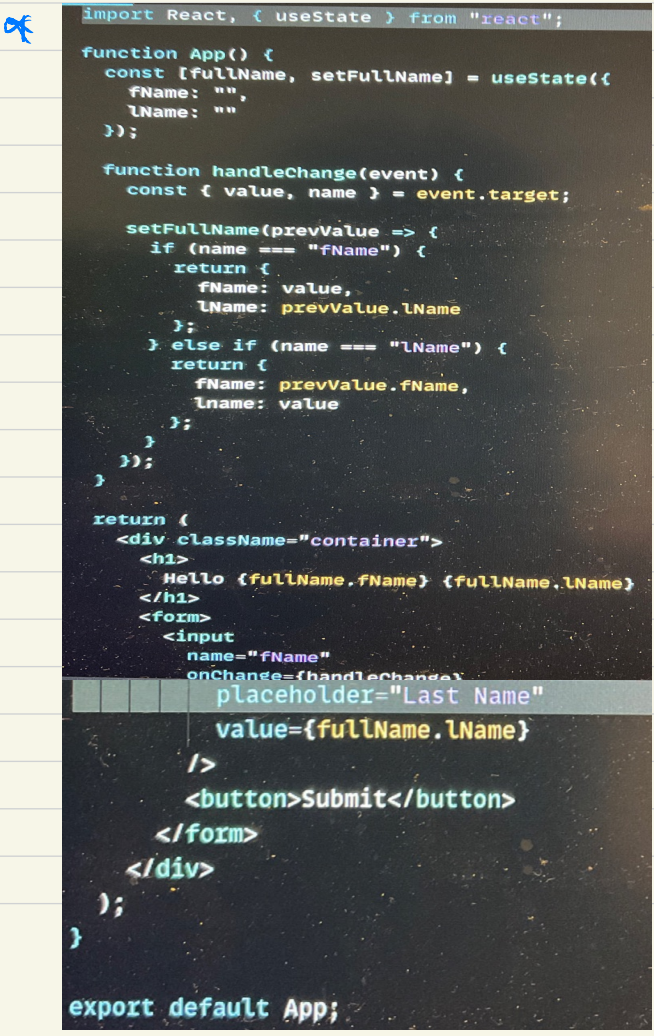






```jsx
import React, {useState} from "react";

function App() {
  let[firstName,setFirstName]=useState("");
    let[lastName,setLastName]=useState("");
    function handleFirstName(event){
      setFirstName(event.target.value);
    }
    function handleLastName(event){
      setLastName(event.target.value);
    }
  return (
    <div className="container">
      <h1>Hello {firstName} {lastName}</h1>
      <form>
        <input onChange={handleFirstName} name="fName" placeholder="First Name" value={firstName} />
        <input onChange={handleLastName} name="lName" placeholder="Last Name" value={lastName}/>
        <button>Submit</button>
      </form>
    </div>
  );
}

export default App;
```

So that value matches with the state



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

function App() {
  const [fullName, setFullName] = useState({
    fName: "",
    lName: ""
  });

  function handleChange(event) {
    const { value, name } = event.target;

    setFullName(prevValue => {
      if (name === "fName") {
        return {
          fName: value,
          lName: prevValue.lName
        };
      } else if (name === "lName") {
        return {
          fName: prevValue.fName,
          lname: value
        };
      }
    });
  }

  return (
    <div className="container">
      <h1>
        Hello {fullName.fName} {fullName.lName}
      </h1>
      <form>
        <input
          name="fName"
          onChange={handleChange}
          placeholder="Last Name"
          value={fullName.lName}
        />
        <button>Submit</button>
      </form>
    </div>
  );
}

export default App;
```

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

function App() {
  const [contact, setContact] = useState({
    fName: "",
    lName: "",
    email: ""
  });
  function handleInput(event) {
    let { name, value } = event.target;
    setContact((prevDetails) => {
      if (name === "lName") {
        return {
          fName: prevDetails.fName,
          lName: value,
          email: prevDetails.email
        };
      } else if (name === "fName") {
        return {
          fName: value,
          lName: prevDetails.lName,
          email: prevDetails.email
        };
      } else if (name === "email") {
        return {
          fName: prevDetails.fName,
          lName: prevDetails.lName,
          email: value
        };
      }
    });
  }

  return (
    <div className="container">
      <h1>
        Hello {contact.fName} {contact.lName}
      </h1>
      <p>{contact.email}</p>
      <form>
        <input
          name="fName"
          placeholder="First Name"
          onChange={handleInput}
          value={contact.fName}
        />
        <input
          name="lName"
          placeholder="Last Name"
          onChange={handleInput}
          value={contact.lName}
        />
        <input
          name="email"
          placeholder="Email"
          onChange={handleInput}
          value={contact.email}
        />
        <button>Submit</button>
      </form>
    </div>
  );
}

export default App;
```

To access this in array syntax

return {
  ...prevDetails,
  [name]: value,
};

Note: if I just wrote name: value; it would add this new key to the object

more shorter code?

setContact(prevDetails => ({...prevDetails, [name]: value}))

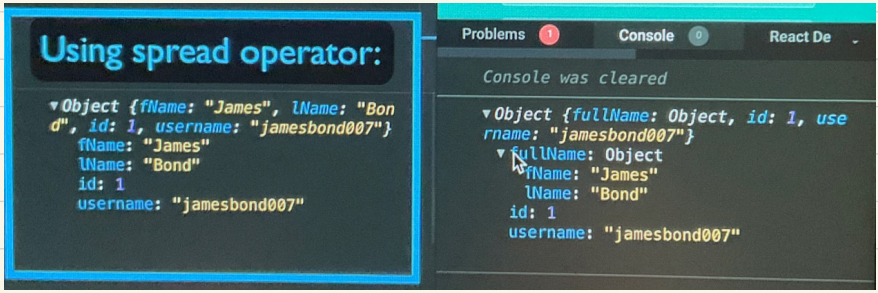
Here bracket is used to return an Object or else without them it won't be interpreted as object in curly brackets {

Spread Operator

* const citrus = ["Lime", "Lemon", "Orange"];
const fruits = ["Apple", "Banana", "Coconut", ... citrus];
// citrus is added to fruits

* const fullName = {
    fName : "James",
    LName : "Bond",
};

const user = {
    ... fullName, (if first fullName)
    id : 1,
    username : "jamesbond001",
};



* Keeper App (Part 2)

```
function App () {
const [input, setInput] = useState ("");
const [item, setItem] = useState ([]);

function handleChange (event) {
 let newValue = event.target.value;
 setInput(newValue);
}
function handleClick ( ) {
  setItem((prevItems)=>([...prevItems, input]));
  setInput ("");
}
<input value={input} onChange={handleChange}
<ul>
   item.map (todoItem => <li>{todoItem}</li>)
<ul>
<button onClick={handleClick}>Add</button>
```

* we create new Value because we can't use event in useState

* we always have to pass the old items with the help of function.

* input area is blank after adding the item

* Delete function

```
function deleteItem (id) {
    setItem ( (prevItems) => {
      return prevItems.filter ((item, index) => {
            return index !== id;
          });
    });
}
```

we will pass function deleteItem in the button with prop onChecked={deleteItem}

```
* function todoItem (props) {
  const [isClicked , setIsClicked ] = useState(false);
  handleClick () {
   return setIsClicked ( prevValue => !prevValue);
  }
 return (
  <div onClick = {handleClick} >
   <li style = {{ textDecoration : isClicked ? "line-through"
    : "none" }} >
     {props. text }
   </li>
  </div>
  );
}
```

opposite

```
items. map((item, index) => {
<TodoItem key={index} text={
  todoItem} id={index} />
})
```

index

* items . map (item , index)

item in the items

* whenever we map through array it is very important to define key for every item of the array

```
import React from "react";

function ToDoItem(props) {
  return (
    <div
      onClick={() => {
        props.onChecked(props.id);
      }}
    >
      <li>{props.text}</li>
    </div>
  );
}

export default ToDoItem;
```

```
import React from "react";

function ToDoItem(props) {
  return (
    <div onClick={props.onChecked(props.id)}>
      <li>{props.text}</li>
    </div>
  );
}
```

Here onChecked will directly be called after rendering this component and will not wait for onClick

* Executed on render

Console

Console was cleare

App.js

```jsx
import React, { useState } from "react";
import ToDoItem from "./ToDoItem";
import InputArea from "./InputArea";

function App() {
  const [items, setItems] = useState([]);

  function addItem(inputText) {
    setItems(prevItems => {
      return [...prevItems, inputText];
    });
  }

  function deleteItem(id) {
    setItems(prevItems => {
      return prevItems.filter((item, index) => {
        return index !== id;
      });
    });
  }

  return (
    <div className="container">
      <div className="heading">
        <h1>To-Do List</h1>
      </div>
      <InputArea onAdd={addItem} />
      <div>
        <ul>
          {items.map((todoItem, index) => (
            <ToDoItem
              key={index}
              id={index}
              text={todoItem}
              onChecked={deleteItem}
            />
          ))}
        </ul>
      </div>
    </div>
  );
}
```

InputArea.js

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

function InputArea(props) {
const [inputText, setInputText] = useState("");
  function handleChange(event) {
    const newValue = event.target.value;
    setInputText(newValue);
  }
  return (
    <div className="form">
      <input onChange={handleChange} type="text" value={inputText} />
      <button onClick={()=>{
        props.onAdd(inputText);
        setInputText("");
      }}>
        <span>Add</span>
      </button>
    </div>
  );
}

export default InputArea;
```

todoitem.js

```jsx
import React from "react";

function ToDoItem(props) {
  return (
    <div
      onClick={() => {
        props.onChecked(props.id);
      }}
    >
      <li>{props.text}</li>
    </div>
  );
}

export default ToDoItem;
```

```jsx
import React, { useState } from "react";
import Header from "./Header";
import Footer from "./Footer";
import Note from "./Note";
import CreateArea from "./CreateArea";

function App() {
  const [notes, setNotes] = useState([]);

  function addNote(newNote) {
    setNotes(prevNotes => {
      return [...prevNotes, newNote];
    });
  }

  function deleteNote(id) {
    setNotes(prevNotes => {
      return prevNotes.filter((noteItem, index) => {
        return index !== id;
      });
    });
  }

  return (
    <div>
      <Header />
      <CreateArea onAdd={addNote} />
      {notes.map((noteItem, index) => {
        return (
          <Note
            key={index}
            id={index}
            title={noteItem.title}
            content={noteItem.content}
            onDelete={deleteNote}
          />
        );
      })}
      <Footer />
    </div>
  );
}
```

App.js

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

function CreateArea(props) {
  const [note, setNote] = useState({
    title: "",
    content: ""
  });

  function handleChange(event) {
    const { name, value } = event.target;

    setNote(prevNote => {
      return {
        ...prevNote,
        [name]: value
      };
    });
  }

  function submitNote(event) {
    props.onAdd(note);
    setNote({
      title: "",
      content: ""
    });
    event.preventDefault();
  }

  return (
    <div>
      <form>
        <input
          name="title"
          onChange={handleChange}
          value={note.title}
          placeholder="Title"
        />
        <textarea
          name="content"
          onChange={handleChange}
          value={note.content}
          placeholder="Take a note..."
          rows="3"
        />
        <button onClick={submitNote}>Add</button>
      </form>
    </div>
  );
}
```

to prevent default behaviour of forms getting auto matically serviced

CreateArea.jsx

Note.jsx

```jsx
import React from "react";

function Note(props) {
  function handleClick() {
    props.onDelete(props.id);
  }

  return (
    <div className="note">
      <h1>{props.title}</h1>
      <p>{props.content}</p>
      <button onClick={handleClick}>DELETE</button>
    </div>
  );
}

export default Note;
```

* npm install @material-ui/core
  npm install @material-ui/icons

* from material-ui we can export prebuild design

Can be used just as components

* floating action button (fab)

* <Zoom in={true}> </Zoom>

* when we refresh page the notes dissapear because React is a frontend library and not responsible for backend