IMY 220 EO

## NPM INSTALLS:

npm install

const fs = require('fs');

npm install fs

const express = require('express');

npm install express

npm install socket.io

# Practicals:

## PRACTICAL 2:

### fileManager.js

const fs = require('fs');

const fileRead = () => {

    const data = fs.readFileSync('events (2).json', 'utf-8');

    return JSON.parse(data);

}

const fileWrite = (*content*) => {

    const jsonData = JSON.stringify(*content*, null, 2);

    fs.writeFileSync('newEvents.json', jsonData, (*err*)=>{

        if(*err*) throw *err*;

    })

}

module.exports = {fileRead, fileWrite};

### dataValidation.js

const checkDate = (*date*) => {

    const eventDate = new Date(*date*);

    const startDate = new Date('2024/09/10');

    const endDate = new Date('2024/09/21');

    return eventDate >= startDate && eventDate <= endDate;

}

const checkName = (*name*) => {

    return /^[a-zA-Z0-9 ]+$/.test(*name*);

}

module.exports = {checkDate, checkName};

### index.js:

const {fileRead, fileWrite} = require('./fileManager');

const {checkDate, checkName} = require('./dataValidation');

const main = () => {

    const events = fileRead();

    const validEvents = events.filter((*event*) => checkDate(*event*.date));

    const updatedEvents = validEvents.map(*event* => {

        return {

            ...*event*,

            validName: checkName(*event*.name)

        }

    })

    fileWrite(updatedEvents);

}

main();

## PRACTICAL 3:

<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8" />

  <title>Practical 3</title>

  <script src="https://unpkg.com/react@18/umd/react.development.js"></script>

  <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"

    integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

  </style>

</head>

<body>

  <div id="root"></div>

*<!-- Your Code goes in this block and only this script tags -->*

  <script type="text/babel">

*// This is provided for you*

    const faqs = [

      { question: "What is React?", answer: "React is a JavaScript library for building user interfaces." },

*// Add your code below this line*

    class App extends React.Component{

      render(){

        return(

          <div>

            <h1> Frequently Asked Questions</h1>

            <Accordion *faqs*={faqs}/>

          </div>

        );

      }

    }

    class Accordion extends React.Component{

      constructor(*props*){

*super*(*props*);

          this.state = {

          toggle: null

        }

        this.toggleQuestion = this.toggleQuestion.bind(this);

      }

      toggleQuestion = (*index*) => {

        this.setState({toggle : this.state.toggle === *index* ? null : *index*});

      }

      render(){

        return(

          <div *className*="accordion" *id*="accordianExample">

              {this.props.faqs.map((*data*, *index*) => (

                <div *className*="accordion-item" *key*={*index*}>

                  <h2 *className*="accordion-header">

                    <button *className*="accordion-button" *type*="button" *onClick*={() => this.toggleQuestion(*index*)}

*aria-expanded*={this.state.toggle === *index*} *aria-controls*={`collapse${*index*}`}>

                      {*data*.question}

                    </button>

                  </h2>

                  <div *id*={`collapse${*index*}`} *className*={`accordion-collapse collapse ${this.state.toggle === *index* ? 'show' : ''}`}

*data-bs-parent*="#accordionExample">

                    <div *className*="accordion-body">

                      {*data*.answer}

                    </div>

                  </div>

                </div>

             ))}

          </div>

        );

      }

    }

    const root = ReactDOM.createRoot(document.getElementById("root"));

    root.render(<App />);

  </script>

</body>

</html>

## Practical 4:

### Package.json:

{

  "name": "prac-4-practice",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "build": "webpack && babel server.js -d dist",

    "start": "npm run build && node dist/server.js",

    "watch": "webpack -w"

  },

  "author": "",

  "license": "ISC",

  "dependencies": {

    "express": "^4.21.1",

    "path": "^0.12.7",

    "react": "^18.3.1",

    "react-dom": "^18.3.1"

  },

  "devDependencies": {

    "@babel/cli": "^7.25.7",

    "@babel/core": "^7.25.7",

    "@babel/preset-env": "^7.25.7",

    "@babel/preset-react": "^7.25.7",

    "babel-loader": "^9.2.1",

    "webpack": "^5.95.0",

    "webpack-cli": "^5.1.4"

  }

}

### Server.js:

const express = require('express');

const app = express();

app.use(express.static("public"));

app.listen(3000, () =>{

    console.log("Server is running on port 3000");

});

### Index.html:

<!DOCTYPE html>

 <html>

    <head>

        <meta charset="utf-8" />

        <title>Node + react</title>

    </head>

    <body>

        <h1>Our first React app in Node</h1>

        <div id="root"></div>

*<!--We’ll get to root when we start adding React elements -->*

        <script type="text/javascript"src="bundle.js"></script>

    </body>

 </html>

### Index.js:

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./App";

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(<App />);

### App.js:

import React from "react";

import RandomMovies from "./components/RandomMovies";

import HelloWorld from "./components/HelloWorld";

const movies = [

    {

        "title": "The Great Adventure",

        "description": "A heartwarming story of friendship and bravery as a group of kids embarks on an unexpected journey through a mysterious forest."

    },

    {

        "title": "Future City",

        "description": "A sci-fi thriller set in a dystopian future where a hacker must battle a corrupt corporation to save humanity."

    },

    {

        "title": "Laugh Out Loud",

        "description": "A hilarious comedy following three best friends who accidentally enter a reality TV show with unexpected consequences."

    },

    {

        "title": "Love and Space",

        "description": "A romantic drama about two astronauts who find love while on a dangerous mission to Mars."

    },

    {

        "title": "The Secret of the Ocean",

        "description": "An animated adventure that follows a young girl and her talking dolphin on a quest to discover the lost city of Atlantis."

    }

]

class App extends React.Component{

    render(){

        return(

            <div>

                <HelloWorld/>

                <RandomMovies *movies*={movies}/>

            </div>

        );

    }

}

export default App;

RandomMovies.js:  
import React from "react";

class RandomMovies extends React.Component{

    constructor(*props*){

*super*(*props*);

    }

    render(){

        return(

            <div>

                <h1>Random Movie List</h1>

                <ul>

                    {this.props.movies.map((*movie*, *index*) => (

                        <li *key*={*index*}>

                            <strong> {*movie*.title}</strong> {*movie*.description}

                        </li>

                    ))}

                </ul>

            </div>

        )

    }

}

export default RandomMovies;

## Practical 5:

### Events.json:

[

    {

        "name": "Picnic in the park",

        "description": "Picnic at a local park",

        "locations": [

            {

                "area": "Brooklyn",

                "date": "2022/10/09",

                "capacity": 25

            },

            {

                "area": "Rosebank",

                "date": "2022/11/06",

                "capacity": 10

            }

        ]

    },

    {

        "name": "Festival",

        "description": "Festival of lights",

        "locations": [

            {

                "area": "Hatfield",

                "date": "2022/11/05",

                "capacity": 15

            },

            {

                "area": "Benoni",

                "date": "2022/11/07",

                "capacity": 30

            }

        ]

    },

    {

        "name": "Pumpkin picking",

        "description": "Pumpkin picking with friends",

        "locations": [

            {

                "area": "Brooklyn",

                "date": "2022/10/18",

                "capacity": 22

            }

        ]

    },

### npm install mongodb

server.js:  
*//Corne de Lange u23788862*

const {MongoClient} = require ('mongodb');

const url = "mongodb+srv://Cornedl:Corne%40mongo@imy220.5pexm.mongodb.net/?retryWrites=true&w=majority&appName=IMY220"

const client = new MongoClient(url);

async function main() {

    try{

        await client.connect();

        console.info("Connected to Mongodb");

        const db = client.db("DBExample"); *//Select DATABASE*

        const collection = db.collection("events"); *//Select Collection*

        const query = await collection.find({"locations": {

            $elemMatch: {

                "area" : "Brooklyn",

                "capacity": {$gt:20},

                "date": {$gt: "2022/10/08", $lt: "2022/10/26"}

        }}}).project({

            "name": 1,

            "description": 1,

            "\_id": 0

        }).toArray();

        console.log(query);

    }

    catch(e){

    }finally {

        await client.close();

    }

}

main().catch(console.error);

### 

# Assignments:

## Assignment 1:

var pets = [

  { name: "Polly", species: "bird", age: 1, adopted: false, adoptedDate: "", adoptionFee: 560 },

  { name: "Fluffy", species: "dog", age: 4, adopted: true, adoptedDate: "2023-03-27",adoptionFee: 890 },

  { name: "Daisy", species: "dog", age: 9, adopted: true, adoptedDate: "2021-01-05", adoptionFee: 780 },

  { name: "Coco", species: "rabbit", age: 3, adopted: true, adoptedDate: "2019-01-30", adoptionFee: 615 },

  { name: "Simba", species: "cat", age: 4, adopted: true, adoptedDate: "2019-09-30", adoptionFee: 995 },

  { name: "Oreo", species: "rabbit", age: 4, adopted: false, adoptedDate: "", adoptionFee: 605 },

  { name: "Bella", species: "cat", age: 6, adopted: false, adoptedDate: "", adoptionFee: 810 },

  { name: "Milo", species: "bird", age: 3, adopted: false, adoptedDate: "", adoptionFee: 740 },

  { name: "Buddy", species: "dog", age: 10, adopted: true, adoptedDate: "2021-02-01", adoptionFee: 735 },

  { name: "Pebbles", species: "bird", age: 4, adopted: false, adoptedDate: "", adoptionFee: 505 },

];

class PetHandler

{

  constructor(*petArray*)

  {

    this.pets = *petArray*;

  }

  findPetsInAgeRange(*minAge*, *maxAge*){

    return this.pets.filter(*pet* => *pet*.age >= *minAge* && *pet*.age <= *maxAge*);

  }

  listAdoptedPetsByDate(){

    return this.pets.filter(*pet* => *pet*.adopted)

    .sort((*a*,*b*) => new Date(*b*.adoptedDate) - new Date(*a*.adoptedDate));

  }

  listPets(...*args*){

    const list = *args*.length ? (Array.isArray(*args*[0]) ? *args*[0] : *args*) : this.pets;

    const createPetItem = *pet* => {

      return *pet*.adopted ?  `${*pet*.name} | ${*pet*.species} | Age: ${*pet*.age} | Adopted!`

      : `${*pet*.name} | ${*pet*.species} | Age: ${*pet*.age}`;

    }

    return list.map(createPetItem);

  }

  calculateUniqueAdoptionFee(...*petNames*){

    const uniqueNames = [...new Set(*petNames*)];

    const uniquePets = this.pets.filter(*pet* => uniqueNames.includes(*pet*.name));

    return uniquePets.reduce((*accumulator*, *pet*) => *accumulator* + *pet*.adoptionFee, 0);

  }

}

Array.prototype.findPetsInAgeRange = function(*minAge*, *maxAge*){

  return this.filter(*pet* => *pet*.age >= *minAge* && *pet*.age <= *maxAge*);

}

Array.prototype.listAdoptedPetsByDate = function(){

  return this.filter(*pet* => *pet*.adopted).sort((*a*,*b*) => new Date(*b*.adoptedDate) - new Date(*a*.adoptedDate));

}

Array.prototype.listPets = function(...*args*){

  const list = *args*.length ? (Array.isArray(*args*[0]) ? *args*[0] : *args*) : this;

  const createPetItem = *pet* => {

    return *pet*.adopted ?  `${*pet*.name} | ${*pet*.species} | Age: ${*pet*.age} | Adopted!`

    : `${*pet*.name} | ${*pet*.species} | Age: ${*pet*.age}`;

  }

  return list.map(createPetItem);

}

Array.prototype.calculateUniqueAdoptionFee = function(...*petNames*){

    const uniqueNames = [...new Set(*petNames*)];

    const uniquePets = this.filter(*pet* => uniqueNames.includes(*pet*.name));

    return uniquePets.reduce((*accumulator*, *pet*) => *accumulator* + *pet*.adoptionFee, 0);

}

## Assignment 2: Scoket.io: index.js:

const socket = io();

socket.on('connect', *message* => {

    console.log('I connected with ID:', socket.id);

});

let poll = document.getElementById('poll');

poll.addEventListener('submit', (*e*) => {

*e*.preventDefault();

    let choice = document.querySelector('input[name="catName"]:checked').value;

    if(choice){

        socket.emit('vote', choice);

    }

});

socket.on('updateVotes', (*poll*) => {

*//\*\*BEFORE LAST TASK\*\*/*

*// const name = catName;*

*// const number = document.querySelector(`label[for="${name}"] span`);*

*// //console.log(number.textContent);*

*// number.textContent = parseInt(number.textContent) + 1;*

*// const totalVotes = document.getElementById('total-votes');*

*// totalVotes.textContent = parseInt(totalVotes.textContent) + 1;*

*//\*\*BEFORE LAST TASK\*\*/*

    let totalVotes = 0;

*poll*.forEach(*vote* => {

        const number = document.querySelector(`label[for="${*vote*.name}"] span`);

        number.textContent = *vote*.votes;

        totalVotes += *vote*.votes;

    });

    document.getElementById('total-votes').textContent = totalVotes;

})

## Server.js:

const express = require('express');

const app = express();

const http = require('http');

const server = http.createServer(app);

const { Server } = require('socket.io');

const io = new Server(server);

const port = 3000;

const Poll = require('./poll.js');

const poll = new Poll([

    { name: 'pebble', votes: 0 },

    { name: 'sunshine', votes: 0 },

    { name: 'miso', votes: 0 },

    { name: 'panko', votes: 0 },

    { name: 'snowball', votes: 0 }

]);

app.get('/', (*req*, *res*) => {

*res*.sendFile(\_\_dirname + '/index.html');

});

app.get('/index.js', (*req*, *res*) => {

*res*.setHeader('Content-Type', 'application/javascript');

*res*.sendFile(\_\_dirname + '/index.js');

});

io.on('connection', (*socket*) => {

    console.log(`A user connected with ID: ${*socket*.id}`);

*socket*.emit('updateVotes', poll.getVotes());

*socket*.on('vote', (*catName*) => {

        poll.vote(*catName*);

        io.emit('updateVotes', poll.getVotes());

    });

*socket*.on('disconnect', () => {

        console.log('A user disconnected');

    });

});

server.listen(port, () => {

    console.log(`Listening on http://localhost:${port}`);

});

Poll.js:  
class Poll {

    constructor(*array*){

        this.cat = *array*;

    }

    vote(*name*){

        return this.cat.filter((*cat*) => *cat*.name === *name*)[0].votes++;

    }

    getVotes(){

        return this.cat;

    }

}

module.exports = Poll;

## Assignment 3:

## 