**Experiment No. 07**

**Title:** To implement data handling with JSON

# Batch: B2 Roll No.: 16010421119 Experiment No.:7

**Aim**: To Implement data handling with JSON.

# Resources needed: Notepad++, Web Browser

**Theory:**

JSON stands for **J**ava**S**cript **O**bject **N**otation. JSON is a **text format** for storing and transporting data. JSON is "self-describing" and easy to understand

* JSON stands for **J**ava**S**cript **O**bject **N**otation
* JSON is a lightweight data-interchange format
* JSON is plain text written in JavaScript object notation
* JSON is used to send data between computers
* JSON is language independent **\***

**Why Use JSON?**

* The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.
* Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.
* JavaScript has a built in function for converting JSON strings into JavaScript objects: **JSON.parse()**
* JavaScript also has a built in function for converting an object into a JSON string: **JSON.stringify()**

Both JSON and XML can be used to receive data from a web server.

# JSON Example

{"employees":[

{ "firstName":"John", "lastName":"Doe" },

{ "firstName":"Anna", "lastName":"Smith" },

{ "firstName":"Peter", "lastName":"Jones" } ]}

# JSON.stringify()

* When sending data to a web server, the data has to be a string.
* Convert a JavaScript object into a string with JSON.stringify().
* Stringify a JavaScript Object

Imagine we have this object in JavaScript:

const obj = {name: "John", age: 30, city: "New York"};

Use the JavaScript function JSON.stringify() to convert it into a string.

const myJSON = JSON.stringify(obj); The result will be a string following the JSON notation. myJSON is now a string, and ready to be sent to a server:

# Example

const obj = {name: "John", age: 30, city: "New York"}; const myJSON = JSON.stringify(obj);

# JSON.parse()

A common use of JSON is to exchange data to/from a web server. When receiving data from a web server, the data is always a string. Parse the data with JSON.parse(), and the data becomes a JavaScript object.

# Example - Parsing JSON

Imagine we received this text from a web server:

'{"name":"John", "age":30, "city":"New York"}'

**Use the JavaScript function JSON.parse() to convert text into a JavaScript object:**

const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');

Make sure the text is in JSON format, or else you will get a syntax error.

**Use the JavaScript object in your page:**

**Example** <p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = obj.name;

</script>

Date objects are not allowed in JSON. If you need to include a date, write it as a string.

You can convert it back into a date object later:

# Example

Convert a String into date

const text = '{"name":"John", "birth":"1986-12-14", "city":"New York"}'; const obj = JSON.parse(text); obj.birth = new Date(obj.birth); document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth;

# Storing Data

When storing data, the data has to be a certain format, and regardless of where you choose to store it, *text* is always one of the legal formats.

JSON makes it possible to store JavaScript objects as text.

# Example Storing data

// Storing data:

const myObj = {name: "John", age: 31, city: "New York"};

const myJSON = JSON.stringify(myObj);

localStorage.setItem("testJSON", myJSON);

// Retrieving data: let text = localStorage.getItem("testJSON"); let obj = JSON.parse(text); document.getElementById("demo").innerHTML = obj.name;

# JSON Server Sending Data

If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

# Example

const myObj = {name: "John", age: 31, city: "New York"}; const myJSON = JSON.stringify(myObj);

window.location = "demo\_json.php?x=" + myJSON;

**Receiving Data**

If you receive data in JSON format, you can easily convert it into a JavaScript object:

# Example

const myJSON = '{"name":"John", "age":31, "city":"New York"}'; const myObj = JSON.parse(myJSON);

document.getElementById("demo").innerHTML = myObj.name;

**JSON HTML HTML Table**

Make an HTML table with data received as JSON:

# Example

const dbParam = JSON.stringify({table:"customers",limit:20});

const xmlhttp = new XMLHttpRequest(); xmlhttp.onload = function() { myObj = JSON.parse(this.responseText); let text = "<table border='1'>" for (let x in myObj) {

text += "<tr><td>" + myObj[x].name + "</td></tr>";

}

text += "</table>" document.getElementById("demo").innerHTML = text;

}

xmlhttp.open("POST", "json\_demo\_html\_table.php");

xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); xmlhttp.send("x=" + dbParam); **HTML Drop Down List**

Make an HTML drop down list with data received as JSON:

# Example

const dbParam = JSON.stringify({table:"customers",limit:20});

const xmlhttp = new XMLHttpRequest(); xmlhttp.onload = function() {

const myObj = JSON.parse(this.responseText);

let text = "<select>" for (let x in myObj) {

text += "<option>" + myObj[x].name + "</option>";

}

text += "</select>"

document.getElementById("demo").innerHTML = text; } }

xmlhttp.open("POST", "json\_demo\_html\_table.php", true);

xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); xmlhttp.send("x=" + dbParam);

**Activity:**

**1. Convert JSON objects into string using JSON.stringify().**

# Replace any data in JSON object JSON.replace()

# 3. Valid JSON string into JSON using JSON.parse()

# Results: (Program printout with output)

**JSON Data: -**

**const data = [**

**{**

**"id": 1,**

**"name": "Aarya Tiwari",**

**"email": "aarya.tiwari@somaiya.edu",**

**"phone": "+917021875752",**

**"age": 19**

**},**

**{**

**"id": 2,**

**"name": "Dhairya Satra",**

**"email": "satra.dv@somaiya.edu",**

**"phone": "+918135692464",**

**"age": 20**

**},**

**{**

**"id": 3,**

**"name": "Pratham Goenka",**

**"email": "pratham.goenka@somaiya.edu",**

**"phone": "+91250175012",**

**"age": 20**

**},**

**{**

**"id": 4,**

**"name": "Dhruv Sharma",**

**"email": "dhruv.sharma@somaiya.edu",**

**"phone": "+910157032149",**

**"age": 34**

**},**

**{**

**"id": 5,**

**"name": "Ameya Yeole",**

**"email": "ameya.yeole@somaiya.edu",**

**"phone": "+91673782012",**

**"age": 21**

**},**

**{**

**"id": 6,**

**"name": "Aditya Awhad",**

**"email": "aditya.awhad@somaiya.edu",**

**"phone": "+919153195688",**

**"age": 90**

**},**

**]**

**CODE: -**

**import React, { useState, useEffect } from 'react';**

**import './RecentCustomers.css'**

**function RecentCustomers() {**

**const stringJSON = JSON.stringify(data) //Converted to String**

**console.log(stringJSON)**

**const dataJSON = JSON.parse(stringJSON) // Converted Back to JSON Data**

**return (**

**<div className='main-table'>**

**<table className='styled-table'>**

**<thead>**

**<tr>**

**<th className='heading'>ID</th>**

**<th className='heading'>Name</th>**

**<th className='heading'>Email</th>**

**<th className='heading'>Phone</th>**

**<th className='heading'>Age</th>**

**</tr>**

**</thead>**

**<tbody>**

**{dataJSON.map(row => (**

**<tr key={row.id}>**

**<td>{row.id}</td>**

**<td>{row.name}</td>**

**<td>{row.email}</td>**

**<td>{row.phone}</td>**

**<td>{row.age}</td>**

**</tr>**

**))}**

**</tbody>**

**</table>**

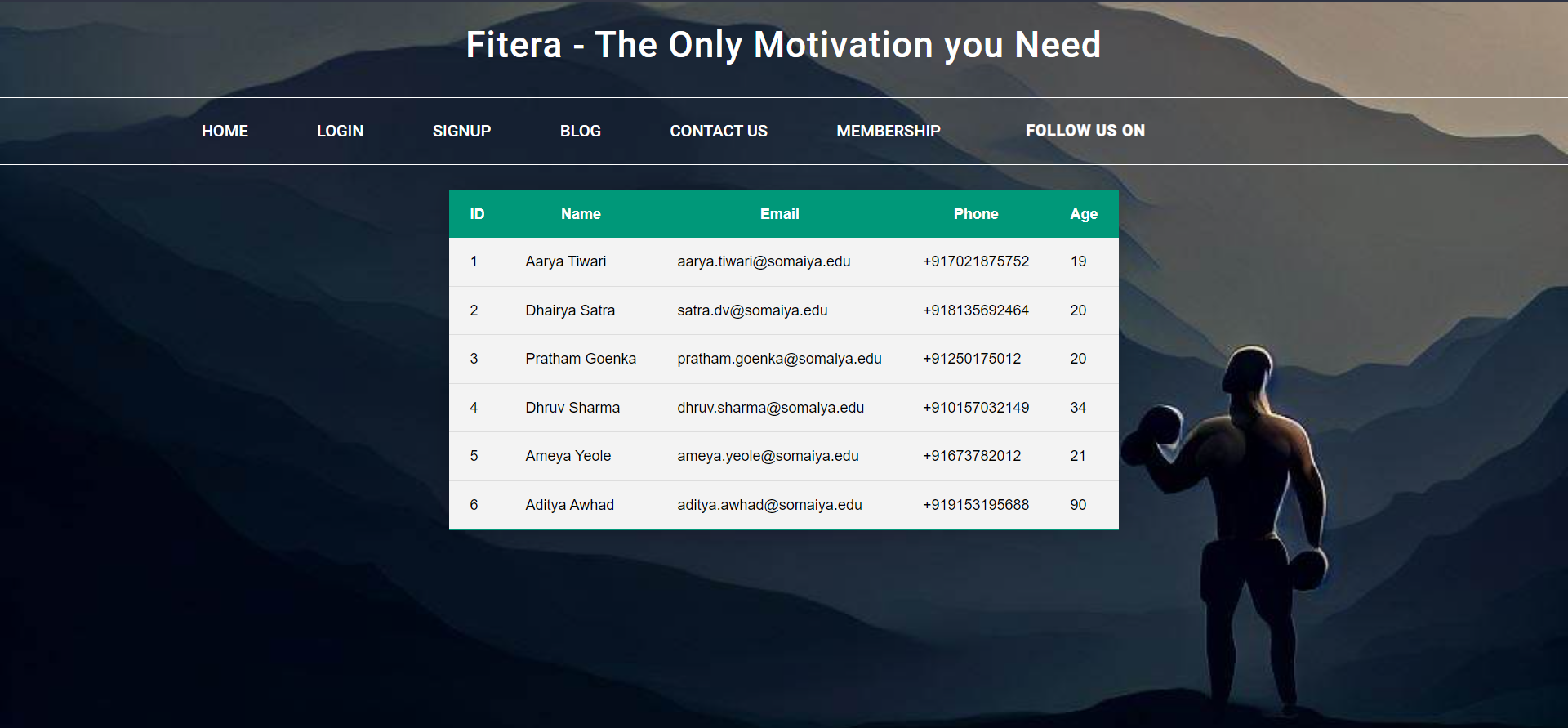
**</div>**

**);**

**}**

**export default RecentCustomers;**

**OUTPUT: -**



**Questions:**

1. Why JSON is better than XML?

Ans : -

Simplicity: JSON is a simpler and more lightweight data format than XML. JSON data can be easily read and understood by both humans and machines, while XML is more complex and requires more parsing and processing.

Size: JSON data is typically smaller in size than equivalent XML data, which can be especially important when transmitting data over a network or storing it on disk.

Readability: JSON data is easier to read and write than XML data. JSON uses a more concise syntax that is similar to JavaScript object notation, which makes it more familiar to many developers.

Native Support: JSON is natively supported in many programming languages and platforms, including JavaScript, Python, and PHP. This makes it easier to work with and integrate into existing systems.

Performance: JSON data is generally faster to parse and process than XML data, due to its simpler syntax and structure. This can be especially important in applications where performance is critical.

1. Write difference between JSON and Javascript

Ans:-

Syntax: JSON has a syntax that is based on a subset of the JavaScript language, but it is not identical to JavaScript. JSON syntax is limited to a small set of data types and structures (such as objects and arrays), and it does not support functions or variables like JavaScript does.

Use: JavaScript is a programming language that is used for creating dynamic web content, such as interactive user interfaces and web applications. JSON, on the other hand, is a data format that is used for transmitting and storing data between different systems.

Data Types: JavaScript supports a wider range of data types than JSON, including numbers, strings, booleans, null, undefined, and more complex data types such as objects and functions. JSON, on the other hand, supports a smaller set of data types, including strings, numbers, booleans, null, objects, and arrays.

Parsing: JavaScript code can be executed directly by a JavaScript interpreter, while JSON data must be parsed and processed by a separate JSON parser. Most programming languages and platforms have built-in support for parsing JSON data.

Object representation: In JavaScript, objects can have methods and functions as properties, whereas in JSON, objects can only have key-value pairs where the values are limited to the supported data types.

**Outcomes:**

**CO3: Apply JavaScript and JSON for web application development**

**Conclusion: (Conclusion to be based on the outcomes achieved)**

**We can conclude that we have learnt about JSON data storage and Retrieval.**

# Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

**References:**

**Books/ Journals/ Websites:**

* “Web technologies: Black Book”, Dreamtech Publications
* [http://www.w3schools.com](http://www.w3schools.com/)