**Experiment No. 07**

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

## Batch: A1 Roll No.: 16010422013 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.sringify().**

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

1. **Valid JSON sting into JSON using JSON.parse() Results: (Program printout with output)**

<!DOCTYPE html>

<html>

<head>

<title>JSON Example</title>

</head>

<body>

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

<script>

*var* myObj = { name: "Shreya", age:20, city: "Mumbai" };

*var* myJSON = JSON.stringify(myObj);

// Replace "Shreya" with "Waingankar" (case-sensitive) myJSON = myJSON.replace("Shreya", "Waingankar");

// Additional objects

*var* myObj2 = { name: "John", age: 25, city: "New York" };

*var* myObj3 = { name: "Alice", age: 30, city: "London" };

*var* myJSON2 = JSON.stringify(myObj2);

*var* myJSON3 = JSON.stringify(myObj3);

*var* myNewObj = JSON.parse(myJSON);

*var* myNewObj2 = JSON.parse(myJSON2);

*var* myNewObj3 = JSON.parse(myJSON3);

document.getElementById("demo").innerHTML = "Name: " + myNewObj.name +

"<br>Age: " + myNewObj.age + "<br>City: " + myNewObj.city + "<br><br>" +

"Name: " + myNewObj2.name + "<br>Age: " + myNewObj2.age + "<br>City: " + myNewObj2.city + "<br><br>" +

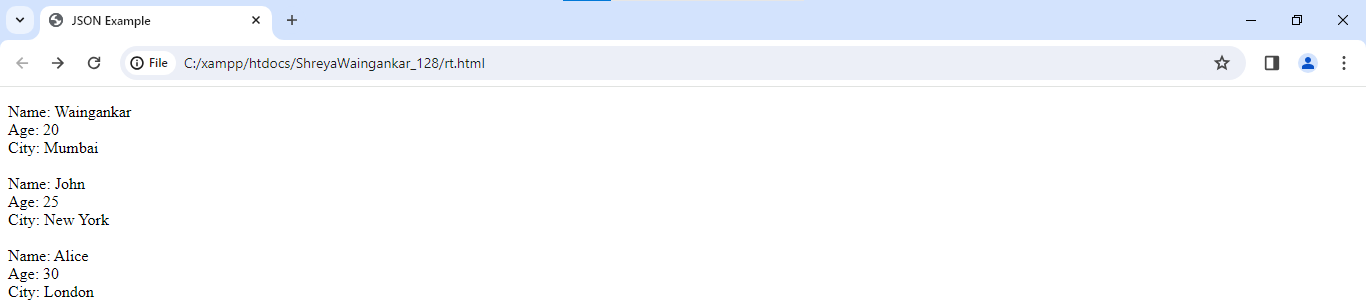
"Name: " + myNewObj3.name + "<br>Age: " + myNewObj3.age + "<br>City: " + myNewObj3.city;

</script>

</body>

</html>

**Output:**



## Questions:

1. Why Jason is better than xml?
2. Write difference between JSON and Javascript

Ans1)

SON (JavaScript Object Notation) and XML (eXtensible Markup Language) are both popular formats used for structuring and storing data. Each has its own advantages and use cases, but JSON is often preferred over XML for several reasons:

## Readability and Simplicity:

* + JSON is generally more human-readable and easier to understand than XML. JSON uses a lightweight, key-value pair format, which closely resembles how data is represented in many programming languages, especially JavaScript.
  + XML, on the other hand, uses tags and attributes, which can make the structure more complex and verbose, especially for nested data.

## Lightweight:

* + JSON tends to be more lightweight in terms of both file size and parsing overhead compared to XML. This is because JSON has a simpler syntax with fewer

characters needed for representing data.

## Native to JavaScript:

* + JSON is a native part of JavaScript, meaning it can be easily parsed and manipulated using built-in functions (**JSON.parse()** and **JSON.stringify()**). This makes JSON integration seamless in JavaScript applications.

## Better for Data Interchange:

* + JSON is often preferred for data interchange between web servers and web clients (such as browsers or mobile apps) due to its lightweight nature and easy parsing capabilities.
  + XML is still widely used in scenarios where document structure and metadata are critical, such as configuration files, markup languages (like HTML), or when integrating with systems that rely heavily on XML.

## Support for Arrays:

* + JSON has built-in support for arrays, making it a natural choice for representing lists of data. In XML, arrays need to be represented using repeating elements or specialized attributes, which can be more cumbersome.

## Performance:

* + JSON parsing and generation are generally faster and consume less memory compared to XML, especially in web environments where bandwidth and processing resources are important factors.

## Integration with Web APIs:

* + JSON has become the de facto standard for many web APIs due to its simplicity and ease of use. Most modern web APIs return data in JSON format, making it convenient for client-side consumption.

While JSON has several advantages over XML, it's important to note that XML still has its place, especially in scenarios where document structure, metadata, or compatibility with legacy systems are priorities. Ultimately, the choice between JSON and XML depends on the specific requirements and constraints of a given project.

Ans2)

JSON (JavaScript Object Notation) and JavaScript are closely related, but they serve different purposes and have distinct characteristics. Here are the key differences between JSON and JavaScript:

## Purpose:

* + JSON: JSON is a lightweight data interchange format used for transmitting data between a server and a web application. It is language-independent and can be used with any programming language.
  + JavaScript: JavaScript is a programming language commonly used for creating dynamic and interactive web pages. It runs in web browsers and on servers (using Node.js) and can be used for a wide range of purposes, including client-side scripting, server-side development, and more.

## Syntax:

* + JSON: JSON syntax is a subset of JavaScript object literal notation. It consists of key-value pairs separated by colons, with keys and strings enclosed in double quotes, and data types including objects, arrays, strings, numbers, booleans, and null.
  + JavaScript: JavaScript syntax is more extensive and includes features such as variables, functions, conditionals, loops, and object-oriented programming constructs. In JavaScript, objects are defined using curly braces **{}**, arrays using square brackets **[]**, and functions using the **function** keyword.

## Usage:

* + JSON: JSON is primarily used for data interchange between a server and a web application. It is commonly used in AJAX requests, APIs, configuration files, and

storing data in NoSQL databases.

* + JavaScript: JavaScript is used for creating dynamic and interactive web pages. It provides functionality for handling user events, manipulating the DOM (Document Object Model), making AJAX requests, validating form data, creating animations, and much more.

## Platform Dependency:

* + JSON: JSON is platform-independent and can be parsed and generated by any programming language with JSON support.
  + JavaScript: JavaScript is a programming language that runs in web browsers and can also run on servers using platforms like Node.js. While JavaScript is widely supported across different web browsers, there may be slight differences in behavior between browsers.

## Data Types:

* + JSON: JSON supports basic data types including objects, arrays, strings, numbers, booleans, and null.
  + JavaScript: JavaScript supports a wider range of data types including objects, arrays, strings, numbers, booleans, functions, dates, regular expressions, and more.

In summary, while JSON and JavaScript share similarities in syntax, they serve different purposes and have distinct features and use cases. JSON is a data interchange format primarily used for transmitting data between a server and a web application, while JavaScript is a versatile programming language used for creating dynamic and interactive web pages.

**Outcomes:**

# CO-4: Implement web application using React JS, Angular JS, JSON and CBOR

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

# Through this experiment, we understood that the JSON (JavaScript Object Notation) and JavaScript are distinct entities with unique roles and characteristics. JSON serves as a lightweight data interchange format, facilitating seamless communication between servers and web applications, while JavaScript powers dynamic and interactive web experiences with its extensive functionality. Despite their close relationship in syntax, JSON's platform- independent nature contrasts with JavaScript's versatility as a full-fledged programming language, each contributing to the rich ecosystem of web development in its own way.

## 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/)