Experiment No. 07

Title: To implement data handling with JSON

Batch: B-4 **Roll No.: 16010422234** **Name: Chandana Ramesh Galgali**

Experiment No.: 7

Aim: To Implement data handling with JSON.

Resources needed: Notepad++, Web Browser

Theory:

JSON stands for **J**avaScript **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**avaScript **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()

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

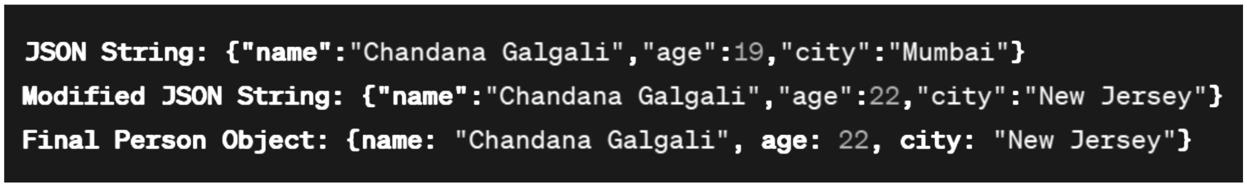
3. Valid JSON string into JSON using JSON.parse()

Results: (Program printout with output)

Program

```javascript
// Activity 1: Convert JSON objects into a string using JSON.stringify()
const person = {
    name: "Chandana Galgali",
    age: 19,
    city: "Mumbai"
};
const personJSON = JSON.stringify(person);
console.log("JSON String:", personJSON);
// Activity 2: Replace any data in JSON object. There's no JSON.replace(),
so we modify the object itself
let personParsed = JSON.parse(personJSON);
personParsed.age = 22;
personParsed.city = "New Jersey";
// Then convert it back to a JSON string
const modifiedPersonJSON = JSON.stringify(personParsed);
console.log("Modified JSON String:", modifiedPersonJSON);
// Activity 3: Convert valid JSON string into JSON using JSON.parse()
const finalPersonObject = JSON.parse(modifiedPersonJSON);
console.log("Final Person Object:", finalPersonObject);
```

Output



```
JSON String: {"name":"Chandana Galgali","age":19,"city":"Mumbai"}
Modified JSON String: {"name":"Chandana Galgali","age":22,"city":"New Jersey"}
Final Person Object: {name: "Chandana Galgali", age: 22, city: "New Jersey"}
```

Questions:

1. Why is json better than xml?

Ans: a) Simplicity: JSON is less verbose and easier to read and write than XML.

b) Data Interchange: JSON is often faster to parse and uses less bandwidth, making it a better choice for web applications.

c) Compatibility: JSON's data types and structure closely align with the way objects are built in many programming languages, particularly JavaScript, facilitating seamless data interchange.

2. Write the differences between JSON and Javascript.

Ans: a) Data Format vs. Programming Language: JSON is a data format (text-based), while JavaScript is a scripting language. JSON uses JavaScript syntax for formatting data, but it is language-independent and can be used with many programming languages.

b) Use Case: JSON is specifically designed for data interchange, whereas JavaScript is used for a wide range of programming tasks, including web development, server-side applications, and more.

Outcomes: Implement web application using React JS, Angular JS, JSON and CBOR

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

The JavaScript code illustrates the core operations involved in handling JSON data within a web development context. It demonstrates how to serialize a JavaScript object into a JSON string, modify the object's properties, and deserialize the JSON string back into a JavaScript object. These processes are essential for effective data interchange between clients and servers in web applications, ensuring data is formatted correctly for transmission and then appropriately parsed for use within application logic. The example showcases the flexibility and utility of JSON for web development, highlighting its role as a vital tool in modern programming environments.

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
