Experiment No. 07

Title: To implement data handling with JSON

Batch:B1 **Roll No.:16010420133** **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().**
2. **Replace any data in JSON object JSON.repalce() 3. Valid JSON sting into JSON using JSON.parse()**

Results: (Program printout with output):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
<center>
    <form name="myName">
        <label for="fname">First name:</label><br>
        <input type="text" id="fname" name="fname"><br><br>
<label for="lname">Last name:</label><br>
        <input type="text" id="lname" name="lname"><br><br>
<label for="favourite_food">sport:</label><br>
        <input type="text" id="sport" name="sport"><br><br>
        <label for="pet_name">achievement:</label><br>
        <input type="text" id="achievement" name="achievement"><br><br>
        <input type="submit" value="Submit" onclick="print()">
    </form>
</center>


    <div class="demo"></div>


    <script>
         function print(){             let formData= {
first_name: form['fname'].value,              last_name:
form['lname'].value,             favourite_food:
form['favourite_food'].value,             pet_name:
form['pet_name'].value,
        }

        Object.entries(formData).forEach(entry => {             let
str = `<br /> ${entry[0]}: ${entry[1]}`
document.getElementsByClassName("demo")[0].innerHTML+= str
        })
        }


    </script>
</body>
</html>
```

Questions:

1. Why Jason is better than xml?

XML is much more difficult to parse than JSON.
JSON is parsed into a ready-to-use JavaScript object.

For AJAX applications, JSON is faster and easier than XML:

Using XML

Fetch an XML document
Use the XML DOM to loop through the document
Extract values and store in variables
Using JSON

Fetch a JSON string
JSON.Parse the JSON string

2. Write difference between JSON and Javascript

JSON	XML
It is JavaScript Object Notation	It is Extensible markup language
It is based on JavaScript language.	It is derived from SGML.

It is a way of representing objects.	It is a markup language and uses tag structure to represent data items.
It does not provides any support for namespaces.	It supports namespaces.
It supports array.	It doesn't supports array.
Its files are very easy to read as compared to XML.	Its documents are comparatively difficult to read and interpret.
It doesn't use end tag.	It has start and end tags.
It is less secured.	It is more secured than JSON.
It doesn't supports comments.	It supports comments.
It supports only UTF-8 encoding.	It supports various encoding.

.

Outcomes:

Apply Javascript and JSON for web application development.

Conclusion: (Conclusion to be based on the outcomes achieved) Implemented data handling with JSON.

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