Title :- Understand and implement JSON

JSON – JavaScript Object Notation

JSON is a syntax for storing and exchanging data.

JSON is text format.

JavaScript Variable vs JavaScript Object

JS variable :- The variable assign simple value or contain only simple value.

Ex : var car = “Vitz”;

JS Object :- Objects are variable too. But object can contain many values.

Ex: var car ={type:"Vitz", color:"White", model:"2018"};

Exchanging Data

When exchanging data between a browser and server, the data can only be text. JSON is text so can convert any JavavScript object into JSON, and send JSON to the server. And also convert any JSON received from the server into JavaScript object. No need complicate parsing and translation process.

Sending Data

The JavaScript object can convert into JSON and send it to a server.

```
EX 01 : var car = {
        type:"Vitz",
        color:"White",
        model:"2018"
        };
        var carDetails = JSON.stringify(car);
        document.getElementById('demo').innerHTML = carDetails;
```

Receiving Data

Received data in JSON format, the data can convert it into a JavaScript object.

```
Ex 02:  var student = '{"name":"John", "age":21, "city":"Colombo"}';
        var JSObject = JSON.parse(student);
        document.getElementById('demo').innerHTML = "Student Name: " + JSObject.name
        + "<br>" + "Age: " + JSObject.age + "<br>" + "City: " + JSObject.city;
```

Storing Data

When storing data, the data has to be a certain format, no need to worry about where to store.

Ex 03:

```
var employee = { name: "John", age: 31, city: "New York" };

var employeeJSON = JSON.stringify(employee);

localStorage.setItem("testing", employeeJSON);
```

Retrieving Data:

```
var text = localStorage.getItem("testing");

var obj = JSON.parse(text);

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

Note 01: JSON is language independent. JSON uses JavaScript syntax, but the JSON format is text only. Text can be read and used as a data format by any programming language.

Note 02: JSON.parse() method change string data into JavaScript object.

JSON Syntax

The JSON syntax is a subset of the JavaScript syntax.

JSON Syntax Rules

JSON syntax is derived from JavaScript object notation syntax.

01. Data is in name/value pairs.

Example: "type":"Vitz"

02. Data is separated by commas.

Example: "type":"Vitz", "model":"2018", "color":"white"

03. Curly braces hold objects.

Example: var car ={ “type”:“Vitz”, “model”:“2018”, “color”:“white”}

04. Square brackets hold arrays.

Example: var car ={ “type”:[“Vitz”, “Fit”, “Aqua”],“model”:“2018”,“color”:“white”}

JSON Data - A Name and a Value

JSON data is written as name/value pairs.

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value.

Example:- "name":"John"

Key Value

Note: JSON keys must be string, written with double quotes. But in JavaScript keys can be String/numbers.

Example:- { name:"John"}

JSON Values

In **JSON**, values must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

In **JavaScript** values can be all of the above, plus any other valid JavaScript expression, including:

- a function
- a date
- undefined

Note 01: JSON uses Javacript Syntex.

Note 02: JSON file extension .json

JSON vs XML

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

Example:-

```
{"employees":[

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

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

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

]}
```

Q1. Write the XML for this JSON example. (https://www.convertjson.com/json-to-xml.htm)

```
<root>
  <employees>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
  </employees>
  <employees>
    <firstName>Anna</firstName>
    <lastName>Smith</lastName>
  </employees>
  <employees>
    <firstName>Peter</firstName>
    <lastName>Jones</lastName>
  </employees>
</root>
```

Similarity between JSON and XML

- Both JSON and XML are "self describing". (human readable)
- Both JSON and XML are hierarchical. (values within values)
- Both JSON and XML can be parsed and used by lots of programming languages.
- Both JSON and XML can be fetched with an XMLHttpRequest.

Different between JSON and XML

- JSON doesn't use end tag.
- JSON is shorter.
- JSON is quicker to read and write
- JSON can use arrays.

Note:- XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

JSON Data Types

In JSON, values must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

JSON Strings

Strings in JSON must be written in double quotes.

Example : {“name”:“John”}

JSON Numbers

Numbers in JSON must be an integer or a floating point.

Example : {“age”:30}

JSON Objects

Values in JSON can be objects.
Example:

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

JSON Arrays

Values in JSON can be arrays.

Example:

```
{
        "employees":[ "John", "Anna", "Peter" ]
}
```

JSON Booleans

Values in JSON can be true/false.

Example : { "Submission":true }

JSON Null

Values in JSON can be null.

Example { "middlename":null }

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.

```
<html>

<body>

<h2>Create Object from JSON String</h2>

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

```
<script>
var txt = '{"name":"John", "age":30, "city":"New York"}'
var obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>
</body>
</html>
```

JSON From the Server

Request JSON from the server by using an AJAX request.
The response from the server is written in JSON format, can parse the string into a JavaScript object.

Try this example:-

First create the below file (Save the file as json_demo.txt)

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

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
    var myObj = JSON.parse(xmlhttp.responseText);
    document.getElementById("demo").innerHTML = myObj.name;
```

```
  }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
</script>
</body>
</html>
```

Array as JSON

When using the JSON.parse() on a JSON derived from an array, the method will return a JavaScript array, instead of a JavaScript object.

Try this example

Create a file it contains json array object.

```
["Ford", "BMW", "Fiat"]
```

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
    var myArr = JSON.parse(xmlhttp.responseText);
    document.getElementById("demo").innerHTML = myArr[0];
  }
```

```
};

xmlhttp.open("GET", "json_array_demo.txt", true);

xmlhttp.send();

</script>

</body>

</html>
```

Parsing Dates

Date objects are not allowed in JSON.

But, we need to include a date in some situations. For that we write it as a string then can convert it back into a **date object.**

Try this example

```
<html>

<body>

<h2>Convert a string into a date object.</h2>

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

<script>

var text = '{"name":"John", "birth":"1986-12-14", "city":"New York"}';

var obj = JSON.parse(text);

obj.birth = new Date(obj.birth);

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

</script>

</body>

</html>
```

Parsing Functions

Functions are not allowed in JSON.

If you need to include a function, write it as a string.

You can convert it back into a function later.

```
<!DOCTYPE html>

<html>

<body>

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

<script>

var text = '{"name":"John", "age":"function() {return 30;}", "city":"New York"}';

var obj = JSON.parse(text);

obj.age = eval("(" + obj.age + ")");

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

</script>

</body>

</html>
```

Note : eval() is a global **function** in JavaScript that evaluates a specified string as JavaScript code and executes it. **eval** can convert string to JSON object.

JSON.stringify()

Convert a JavaScript object into a string

```
<html>

<body>

<h2>Create JSON string from a JavaScript object.</h2>

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

```
<script>

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

var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;

</script>

</body>

</html>
```

Stringify a JavaScript Array

It is also possible to stringify JavaScript arrays.

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

<script>

var arr = [ "John", "Peter", "Sally", "Jane" ];

var myJSON = JSON.stringify(arr);

document.getElementById("demo").innerHTML = myJSON;

</script>
```

Stringify Dates

In JSON, date objects are not allowed. The JSON.stringify() function will convert any dates into strings.

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

<script>

var obj = { name: "John", today: new Date(), city: "New York" };

var myJSON = JSON.stringify(obj);
```

```
document.getElementById("demo").innerHTML = myJSON;

</script>
```

Stringify Functions

In JSON, functions are not allowed as object values.

The JSON.stringify() function **will remove any functions from a JavaScript object**, **both the key and the value.**

Example:-

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

<script>

var obj = { name: "John", age: function () {return 30;}, city: "New York" };

var myJSON = JSON.stringify(obj);

document.getElementById("demo").innerHTML = myJSON;

</script>
```

JSON Objects

JSON objects are surrounded by curly braces {}.

JSON objects are written in key/value pairs.

Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).

Keys and values are separated by a colon.

Each key/value pair is separated by a comma.

Example:-

```
{ "name":"John", "age":30, "car":null }
```

Accessing Object Values

- It can access the object values by using dot (.) notation.

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

<script>

var myObj, x;

myObj = {"name":"John", "age":30, "car":null};

x = myObj.name;

document.getElementById("demo").innerHTML = x;

</script>
```

- It can also access the object values by using bracket ([]) notation.

```
x = myObj["name"]
```

Looping an Object

It can loop through object properties by using the for-in loop.

EX11.

```
<script>

var myObj, x;

myObj = {"name":"John", "age":30, "car":null};

for (x in myObj) {

  document.getElementById("demo").innerHTML += x + "<br>";

}

</script>
```

In a for-in loop, use the bracket notation to access the propertyvalues.

```
<script>
var myObj, x;
myObj = {"name":"John", "age":30, "car":null};
for (x in myObj) {
  document.getElementById("demo").innerHTML += myObj[x] + "<br>";
}
</script>
```

Modify Value

Can use the dot notation to modify any value in a JSON object.

EX13

```
<script>
var myObj, i, x = "";
myObj = {
  "name":"John",
  "age":30,
  "cars": {
  "car1":"Ford",
  "car2":"BMW",
  "car3":"Fiat"
  }
}
myObj.cars.car2 = "Mercedes";
```

```
for (i in myObj.cars) {
  x += myObj.cars[i] + "<br>";
}

document.getElementById("demo").innerHTML = x;
</script>
```

Delete Object Properties

Use the delete keyword to delete properties from a JSON object.

EX14

```
<p id="demo"></p>
<script>
var myObj, i, x = "";
myObj = {
  "name":"John",
  "age":30,
  "cars": {
  "car1":"Ford",
  "car2":"BMW",
  "car3":"Fiat"
  }
}
delete myObj.cars.car2;
```

```
for (i in myObj.cars) {
  x += myObj.cars[i] + "<br>";
}

document.getElementById("demo").innerHTML = x;
</script>
```

JSON Arrays

Arrays can be values of an object property.

Example:-

```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

Accessing Array Values

Access the array values by using the index number.

```
x = myObj.cars[0]
```

Nested arrays in JSON Objects

Values in an array can also be another array, or even another JSON object.

```
myObj = {
  "name":"John",
  "age":30,
  "cars": [
    { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
    { "name":"BMW", "models":[ "320", "X3", "X5" ] },
    { "name":"Fiat", "models":[ "500", "Panda" ] }
  ]
 }
```

Try this example

EX15

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

<script>

var myObj, i, j, x = "";

myObj = {

  "name":"John",

  "age":30,

  "cars": [

    {"name":"Ford", "models":["Fiesta", "Focus", "Mustang"]},

    {"name":"BMW", "models":["320", "X3", "X5"]},

    {"name":"Fiat", "models":["500", "Panda"] }

  ]

}

for (i in myObj.cars) {

  x += "<h2>" + myObj.cars[i].name + "</h2>";

  for (j in myObj.cars[i].models) {

    x += myObj.cars[i].models[j] + "<br>";

  }

}

document.getElementById("demo").innerHTML = x;

</script>
```

Exercise - HTML Table

Create an HTML table with data received as JSON.

Step 1.Create a JSON file.

Step 2. Create a html file with table and using XMLHttpRequest to load the Json data.

Step 3. Display the data in the html table.