Introduction

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is "self-describing" and easy to understand
- JSON is language independent
- JSON uses JavaScript syntax, but the JSON format is text only.

Advantage:it can easily be sent to and from a server, and used as a data format by any programming language.

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

Advantages of JSON

1. **JSON is Faster** -- its syntax is very small and light weighted that's the reason that it executes the response in the faster way.

2. **Schema Support-**--It has the wide range of supported browser compatibility with the operating systems so the applications made with the coding of JSON doesn't require much effort to make it all browser compatible.

3. **Server Parsing** --On the server side parsing is the important part that developers want if the parsing will be fast on the server side then the only user can get the fast response of their response

4. **Tool for sharing data** -- JSON is the best tool for the sharing data of any size even audio, video etc. This is because JSON stores the data in the arrays so data transfer makes easier. For this reason, JSON is a superior file format for web APIs and for web development.

Limitations of JSON

- First and foremost, in JSON has no error handling for JSON calls. If the dynamic script insertion works, you get called and will get the response perfectly. If not inserted, nothing happens. It just fails silently. For example, you are not able to catch a 404 error from the server, Nor can you cancel or restart the request. You can, however, timeout after waiting a reasonable amount of time.
- Another major drawback of JSON is that it can be quite dangerous if used with untrusted services or untrusted browsers, because a JSON service returns a JSON response wrapped in a function call, which will be executed by the browser if it will be used with untrusted browser it can be hacked, this makes the hosting Web Application Vulnerable to a variety of attacks.

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

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Receiving Data

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

Example

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

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 in local storage

- //Storing data:
  myObj = {name: "John", age: 31, city: "New York"};
  myJSON = JSON.stringify(myObj);
  localStorage.setItem("testJSON", myJSON);

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

JSON Syntax

- The JSON syntax is a subset of the JavaScript syntax.

<u>JSON Syntax Rules:</u>

1. Data is in name/value pairs
2. Data is separated by commas
3. Curly braces hold objects
4. Square brackets hold arrays

Working with JSON Objects

- Accessing Object Values
- Looping an Object
- Nested JSON Objects [Refer to example program]
- Modify Values[refer to example program]
- Delete Object Properties

Nested JSON object

- myObj = {
  "name":"John",
  "age":30,
  "cars": {
  "car1":"Ford",
  "car2":"BMW",
  "car3":"Fiat"
  }
  }

Note: You can access nested JSON objects by using the dot notation or bracket notation

```
x = myObj.cars.car2;
//or:
x = myObj.cars["car2"];
```

JSON.parse()

- Example - Parsing JSON

Imagine we received this text from a web server:

'{ "name":"Jhanvi", "age":20, "city":"New Delhi"}’

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

```
var obj = JSON.parse('{ "name":"Jhanvi", "age":20, "city":"New Delhi"}');
```

JSON.stringify()

- Imagine we have this object in JavaScript:

var obj = { name: "Jai", age: 30, city: “Jaipur" };

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

var myJSON = JSON.stringify(obj);

Conversion of JSON object to string

- Refer to Programs

Conversion of string to JSON object

Refer to programs

JSON Array

1.The square brackets [ ] are used to declare JSON array.
2.JSON array are ordered list of values.
3.JSON array can store multiple value types.
4.JSON array can store string, number, boolean, object or other array inside JSON array.
5.In JSON array, values must be separated by comma.
6.Arrays in JSON are almost the same as arrays in JavaScript.

e.g.

```
{
    "name" : "Admin",
    "age" : 36,
    "rights" : [ "admin", "editor", "contributor" ]
}
```

How to access array values in JSON

- Arrays in JSON are almost the same as arrays in JavaScript.
- In JSON, array values must be of type string, number, object, array, Boolean or *null*. Arrays can be values of an object property.

→You access the array values by using the index number.

→You can access array values by using a for-in loop.

→Or you can use a for loop.

→Values in an array can also be another array, or even another JSON object(nested array in JSON Object).

→ example program for better understanding follows next:

```
<!DOCTYPE html>
<html>
<body>

<p>Access an array value of a JSON object.</p>

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

<script>

var myObj, x;
myObj = {
   "name":"John",
   "age":30,
   "cars":[ "Ford", "BMW", "Fiat" ]
};
x = myObj.cars[0];
document.getElementById("demo").innerHTML = x;

</script>

</body>
</html>
```

Modifying the value present at an index in array

- Array value of a JSON object can be modified. It can be simply done by modifying the value present at a given index.
- If value is modified at an index which is out of the array size, then the new modification will not replace anything in the original information but rather will be an add-on.

Get value at specific index location in array

- You can access the array values by using the index number:

```
x = myObj.rights[0];

//Output

admin
```

Describe Data types which are supported by JSON?

S. No.	Type & Description
1	**Number** double- precision floating-point format in JavaScript
2	**String** double-quoted Unicode with backslash escaping
3	**Boolean** true or false
4	**Array** an ordered sequence of values
5	**Value** it can be a string, a number, true or false, null etc
6	**Object** an unordered collection of key:value pairs
7	**Whitespace** can be used between any pair of tokens
8	**null** empty

Number

- It is a double precision floating-point format in JavaScript and it depends on implementation.
- Octal and hexadecimal formats are not used.
- No NaN or Infinity is used in Number.

Syntax

var json-object-name = { string : number_value, ...}

The following table shows the number types –

S. No.	Type & Description
1	**Integer** Digits 1-9, 0 and positive or negative
2	**Fraction** Fractions like .3, .9
3	**Exponent** Exponent like e, e+, e-, E, E+, E-

Write the syntax of the keyword which is used to delete an item from array in JSON?

- Use the delete keyword to delete items from an array:

```
delete myObj.rights[1];
```

Write the syntax for updating array value at index location.

- Use the index number to modify an array:

```
myObj.rights[1] = "blogger";
```

What is multi-dimensional array in JSON.

We can store array inside JSON array, it is known as array of arrays or multidimensional array.

```
{
     "name": "Admin",
     "age": 36,
     "rights": [
          {"roleName": "admin", "roleIds": [1,2,3]},
          {"roleName": "editor", "roleIds": [4,5,6]},
     ] {"roleName": "contributor", "roleIds": [7,8,9]}
}
```