JSON – Java-Script Object Notation

→ It is a data Interchange format.

→ It is a text format used to exchange data b/w platforms.

→ Parallely, other data interchange formats exists such as XML, CSV etc.

→ JSON is programming language independent, though the name originate from JavaScript ~~still JSON is a portable lang~~ it does not require an extensive knowledge of Javascript. The concentration should be on notations used in JSON.

→ It basically acts as a translator ~~use~~ to interpret data b/w two different s/ms.

JSON Syntax

→ JSON is based on Javascript Object Notation.

→ In JavaScript an object contain both data (property) & functions (methods). But JSON is majorly ~~used for~~ deals with data interchange hence, it only contain data (property) and omits functions.

→ JSON uses Name-value pair as in Javascript

JSON object syntax

→ Its an unordered set of name/value pairs

→ It begins with { (left brace)

→ ends with } (right brace)

→ In Name-value pair, name & value are seperated by colon (:)

→ Each Name/value pairs are separated by comma (,)

Eg: Var empdata = {

"emp_id" : 12345,
"name" : "XYZ",
"location" : "Bangalore".
};

Note: Name property is always enclosed in double quote.

Arrays in JSON

→ They are the ordered collection of values

→ It begin with left bracket ([), ends with right bracket (])

• Name/value pairs are seperated by comma (,)

Eg: Var emp = {

"Emp-id" : 1234,

"name" : "xyz",

"dept_worked" : [24, 25, 26, 27]

};

Json datatypes

1) String (always put in double quotes)

2) Number
   - Integer
   - Real / floating-point number
   - scientific – fixed point number.
   - No octal or hex
   - No NaN or Infinity – uses null instead.

3) Boolean
   - true
   - false

4) Null – specifies no value.

5) Object → unordered key/value pairs wrapped in { }, nesting of object.

6) Arrays → ordered key/value pairs wrapped in [ ]

Applications of JSON.

When & how to use JSON.

1) Transfer data to & from a server
2) Perform asynchronous data calls without requiring a page refresh.
3) Working with data stores.
4) Compile and save form or user data for local storage.

Syntax validation

→ JSON Formatter & Validator
→ JSON Editor online
→ JSON Lint

XML Vs JSON

→ JSON objects has a type & XML is typeless.
→ JSON types: string, number, array, Boolean, XML type: string.
→ JSON has no display capabilities. XML has display abilities.

JSON Schema

→ JSON schema are the means through with the JSON objects can be validated.

→ JSON schema validates the data & then the validated data can be passed to the application program. Thereby reducing the overhead on application programs to check the received data.

→ Following steps to be followed to define a schema.

i) Declare the schema & link it to the draft

→ "$schema": "http://json-schema.org/draft-04/schema#"

ii) Provide title for the name-value pair

→ "title": "RVCE"

iii) Define the property of given title.

→ "title": "RVCE"
"Properties": {
"name": { "type": "string" },
"type": { "type": "string" }
}

→ Schema checks for the following

i) Are data types & the values correct?
ii) Does this include the required data?
iii) Are the data in required format?

JSON schema validation.

→ JSON schema Lint

→ JSON schema Validator

A complete example of using JSON object & its content displayed in required format.

```
<html>
  <head>
    <title> convert JSON Data to HTML Table </title>
    <style>    // formating using css
      th, td, P, input {
          font :14px Verdana;
      }
      Table, th, td
        {
          border : solid 1px #DDD;
          border-collapse : collapse;
          padding : 2px 3px ;
          text-align : center;
        }
      th {
          font-weight : bold;
        }
    </style>
  <script>
      function CreateTableFromJSON ()
        {
```

```
{
  var students = [
        {
          "stud_id" : "01",
          "name" : "xyz",
        }
          :
    ]

Var tableBody = '<table border ="1" ><tr><td> Stud_id </td>
        <td> stud Name </td></tr>'
    students.forEach(function(d)
      { tableBody += '<tr><td>'+ d.stud_id+'</td><td>'
      + d.name +'</td></tr>';
      });
       tableBody += '</table>';
    var divcont = document.getElementById ("showData");
      divcont.innerHTML = tableBody;
    }
</script>
</head>
<body>
  <input type ="button" onclick ="createTableFromJSON()"
    value ="create table from JSON"/>
    <p id ="showData"> </p>
</body>
</html>
```

JSON Parse

→ JSON is used to exchange data to/from a web server.

→ when data is received from web server it is in string format.

→ JSON.parse() is a javascript function used to convert the text/string data into required object format (javascript object).

Eg:

```
<html>
 <head>
    <title> JSON Example </title>
 </head>
 <body>
    <script type = "text/javascript">
        let emp = '[
            {"name" : "PNP",
             "Age" : "23"
            }
        ]'
 console.log (JSON.parse (emp[0].name)
 </script>
 </body>
</html>.
```