JSON

Overview and parsing

What is JSON

- JSON: **J**ava**S**cript **O**bject **N**otation.
- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.
- Commonly used in Web as a vehicle to describe data being sent between systems

Exchanging Data

- When exchanging data between a browser and a server, the data can only be text.
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

JSON Syntax Rules

- JSON syntax is derived from JavaScript object notation syntax:
- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

JSON example

- Despite the name, JSON is a (mostly) **language-independent** way of specifying objects as name-value pairs

```
{
        "skillz": {
                "web": [{
                        "name": "html",
                        "years": 5
                }, {
                        "name": "css",
                        "years": 3
                }],
                "database": [{
                        "name": "sql",
                        "years": 7
                }]
        }
}
```

JSON vs XML

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

```
{"employees":[
    { "firstName":"Srinivas", "lastName":”Jandhyala" },
    { "firstName":“Ruchi", "lastName":"Sharma" },
    { "firstName":“Imran", "lastName":“Khan" }
]}
```

JSON vs XML

XML Example

```
<employees>
   <employee>
      <firstName>Srinivas</firstName> <lastName>Jandhyala</lastName>
   </employee>
   <employee>
      <firstName>Ruchi</firstName> <lastName>Sharma</lastName>
   </employee>
   <employee>
      <firstName>Imran</firstName> <lastName>Khan</lastName>
   </employee>
</employees
```

JSON vs XML

JSON is **Like** XML Because

- 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

Parsing JSON

- When we parse JSON, it means we are converting the string into a JSON object by following the specification, where we can subsequently use in whatever way we want.

- For example in javascript

```
var jsonStr = ‘{“name”: “Ritwik”}’;
var obj = JSON.parse(jsonStr);
console.log(obj.name); // prints “Ritwik”
```

Parsing JSON

- Before parsing, it is just a regular string - we cannot access the data encoded inside.
- After parsing, it becomes a Javascript object where u can access the various data within.

JSON vs XML

JSON is **Unlike** XML Because

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

- ***The biggest difference is: XML has to be parsed with an XML parser (DOM, SAX, Xerces etc.)***
- ***JSON can be parsed by a standard JavaScript function (JSON.parse(jsonString)).***

Valid 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*

Invalid data types

- JSON values **cannot be** one of the following data types:
- a function
- a date
- *undefined*

JSON Strings, Numbers, Objetcs

- **Strings** in JSON must be written in double quotes.

```
{ "name":"Srinivas" }
```

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

```
{ "age":30 }
```

- Values in JSON can be **objects.**

```
{
"employee":{ "name":"Srinivas", "age":30, "city":
"Hyderabad" }
}
```

JSON Arrays, Booleans, null

- Values in JSON can be **arrays**.

```
{
"employees":[ "Srinivas", “Ruchi", "Imran" ]
}
```

- Values in JSON can be **true/false**.

```
{ "sale":true }
```

- Values in JSON can be **null**.

```
{ "middlename":null }
```

JSON Objects

- { "name":"Srinivas", "age":30, "car":null }

- 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.

Accessing Object Values

- Can access the object values by using dot (.) notation:

```
myObj = { "name":"Srinivas", "age":30, "car":null };
x = myObj.name;
```

- Can also access the object values by using bracket ([]) notation:

```
myObj = { "name":"Srinivas", "age":30, "car":null };
x = myObj["name"];
```

Nested JSON Objects

- Values in a JSON object can be another JSON object.
- myObj = {

```
"name":"Srinivas",
"age":30,
"cars": {
    "car1":"Audi",
    "car2":"BMW",
    "car3":"Benz"
}
}
```

Nested JSON Objects

- Access nested JSON objects by using the dot notation or bracket notation:

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

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

- ```
  myObj.cars.car2 = "Opel";
  ```

Delete Object Properties

- **delete** myObj.cars.car2;

JSON Arrays

- ["Audi", "BMW", "Benz"]
- In JSON, array values must be of type string, number, object, array, boolean or *null*.

- Arrays in JSON Objects

- myObj = {
  "name":"Srinivas",
  "age":30,
  "cars":["Audi", "BMW", "Benz"]
  }}

JSON Arrays

- 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":"Srinivas",
    "age":30,
    "cars": [
       { "name":"Audi", "models":[ "Q3", “Q8", "A2" ] },
       { "name":"BMW", "models":[ "320", "X3", "X5" ] },
       { "name":"Benz", "models":[ "E-Class", "G-Class", "GL-Class" ] }
    ]
 }
```

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.

JSON.parse()

- Imagine we received this text from a web server:

  '{ "name":"Srinivas", "age":30, "city":"Hyderabad"}'

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

  var obj = JSON.parse('{ "name"Srinivas", "age":30, "city":"Hyderabad"}');

JSON From the Server

- Can request JSON from the server by using an AJAX request
- As long as the response from the server is written in JSON format, can parse the string into a JavaScript object.

JSON From the Server

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myObj.name;
    }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
```

No.	JSON	XML
1)	JSON stands for JavaScript Object Notation.	XML stands for eXtensible Markup Language.
2)	JSON is simple to read and write.	XML is less simple than JSON.
3)	JSON is easy to learn.	XML is less easy than JSON.
4)	JSON is data-oriented.	XML is document-oriented.
5)	JSON doesn't provide display capabilities.	XML provides the capability to display data because it is a markup language.
6)	JSON supports array.	XML doesn't support array.
7)	JSON is less secured than XML.	XML is more secured.
8)	JSON files are more human readable than XML.	XML files are less human readable.
9)	JSON supports only text and number data type.	XML support many data types such as text, number, images, charts, graphs etc. Moreover, XML offers options for transferring the format or structure of the data with actual data.

No.	JSON	XML
1)	JSON stands for JavaScript Object Notation.	XML stands for eXtensible Markup Language.
2)	JSON is simple to read and write.	XML is less simple than JSON.
3)	JSON is easy to learn.	XML is less easy than JSON.
4)	JSON is data-oriented.	XML is document-oriented.
5)	JSON doesn't provide display capabilities.	XML provides the capability to display data because it is a markup language.
6)	JSON supports array.	XML doesn't support array.
7)	JSON is less secured than XML.	XML is more secured.
8)	JSON files are more human readable than XML.	XML files are less human readable.
9)	JSON supports only text and number data type.	XML support many data types such as text, number, images, charts, graphs etc. Moreover, XML offers options for transferring the format or structure of the data with actual data.