Introduction to JSON

Definition

- JSON stands for JavaScript Object Notation. JSON objects are used for transferring data between server and client.

Introduction

- JSON: JavaScript Object Notation.
- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.

Features of JSON

- It is light-weight
- It is language independent
- Easy to read and write
- Text based, human readable data exchange format

Why use JSON

- Standard Structure: As we have seen so far that JSON objects are having a standard structure that makes developers job easy to read and write code, because they know what to expect from JSON.
- Light weight: When working with AJAX, it is important to load the data quickly and asynchronously without requesting the page re-load. Since JSON is light weighted, it becomes easier to get and load the requested data quickly.
- Scalable: JSON is language independent, which means it can work well with most of the modern programming language. Let’s say if we need to change the server side language, in that case it would be easier for us to go ahead with that change as JSON structure is same for all the languages.

Syntax of JSON

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

Example of JSON

Sending Data

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

Receiving Data

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

JSON vs XML

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

The following JSON and XML examples both define an employees object, with an array of 3 employees:

JSON example

```
{"employees":[ { "firstName":"John", "lastName":"Doe" }
, { "firstName":"Anna", "lastName":"Smith" }, { "firstN
ame":"Peter", "lastName":"Jones" }]}
```

JSON vs XML

XML example

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

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