JSON (JavaScript Object Notation)

JSON - Introduction

- ✓ JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write.
- ✓ It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language.
- ✓ 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.
- ✓ JSON is "self-describing" and easy to understand
- ✓ JSON is language independent *

Why JSON?

JSON is faster and easier than XML when you are using it in AJAX web applications:

Steps involved in exchanging data from web server to browser involves:

Using JSON

Fetch a JSON string

Parse the JSON string using eval() or parse() javascript functions

Using XML

Fetch an XML document from web server

Use the XML DOM to loop through the document

Extract values and store in variables

It also involves type conversions

JSON Syntax Rules

JSON Syntax Rules

✓ Data is in name/value pairs

✓ Data is separated by commas

✓ Curly braces hold objects

✓ Square brackets hold arrays

Example:

The JSON format is almost identical to JavaScript objects.

JSON Keys:

In JSON, keys must be strings, written with double quotes:

"name":"testName"

JavaScript Keys:

In JavaScript, keys can be strings, numbers, or identifier names:

{ name:"testName" }

JSON names require double quotes. JavaScript names don't.

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

JavaScript Values:

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

a string

a number

an object (JSON object)

an array

a boolean

null

a function

a date

undefined

JSON Data Manipulation

Sending Data:

If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

```
var myObj = { "name":"testName", "age":28, "city":"Bangalore" };

var myJSON = JSON.stringify(myObj);

console.log(myJSON);
```

Receiving Data:

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

```
var myJSON = '{ "name":"testName", "age":28,
"city":"testCity" }';

var myObj = JSON.parse(myJSON);

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

Storing Data:

JSON makes it possible to store JavaScript objects as text.

```
//Storing data:

myObj = { "name":"testName", "age":28, "city":"testCity" };

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 Data Types

Valid Data Types

✓ a string

✓ a number

✓ an object (JSON object)

✓ an array

✓ a boolean

✓ null

Strings

{ "name":"testName" }

Numbers

{ "age":28 }

Object

{ "name":"testName", "age":28, "car":null }

Arrays

["Car", "Bike"]

Booleans

{ "productAvailable":true }

null

{ "middleName":null }

Accessing Object Values

myObj = { "name":"testName", "age":28, "car":null };

x = myObj.name; (**OR**)

x = myObj[“name"];

Looping an Object

myObj = { "name":"John", "age":28, "car":null };

```
for (x in myObj) {
    document.getElementById("demo").innerHTML += x+”—“+myObj[x];;
}
```

Nested JSON Objects

```
myObj = {
   "name":"testName",
   "age":30,
   "vehicle": {
      "vehicle1":"Car",
      “vehicle2":"Bike"
    }
 }
```

x = myObj.vehicle.vehicle2; (OR)

x = myObj.vehicle["vehicle2"];

JSON Arrays

Arrays in JSON Objects

```
{ "name":"testName",
"age":30,
"vehicle": [ "Car", “Bike"]    // Array
}
```

Accessing Array Values

```
myObj = { "name":"testName",
   "age":30,
   "vehicle": {
      "vehicle1":"Car",
      “vehicle2":"Bike"
    }
 }
x = myObj.vehicle[0];
```

Looping Through an Array

```
for (i in myObj.vehicle) {
   x += myObj.vehicle[i];
}
```

(OR)

```
for (i = 0; i < myObj.vehicle.length; i++) {
   x += myObj.vehicle[i];
}
```

Nested Arrays in JSON Objects

```
myObj = {
   "name":"testName",
   "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 += "<h1>" + myObj.cars[i].name + "</h1>";
   for (j in myObj.cars[i].models) {
      x += myObj.cars[i].models[j];
   }
}
```

Modify Array Values

```
myObj.cars[1] = "Mercedes";
```

Delete Array Items

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

JSON.parse()

Parsing JSON

We received this text from a web server:

'{ "name":"testName", "age":30, "city":"testCity”}’

```
var obj = JSON.parse('{ "name":"testName", "age":30, "city":"testCity"}');

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

Browser Support

Firefox 3.5

Internet Explorer 8

Chrome

Opera 10

Safari 4

JSON From the Server

Use the XMLHttpRequest to get data from the server:

```
//responseText = { "name":"testName", "age":30, "city":"testCity”}
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        myObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myObj.name; // Output: testName
    }
};
```

Array as JSON

Use the XMLHttpRequest to get data from the server:

```
//responseText = [ "Car", "Bike"]
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        myArr = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myArr[0];
    }
};
```

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

Parsing Dates & Functions

Parsing Dates

Convert a string into a date:

```
var text = '{ "name":"testName", "birth":"1986-12-14",
city":"testCity"}';

var obj = JSON.parse(text);

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

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

Parsing Functions

Functions are not allowed in JSON.

Include function as a string.

Convert a string into a function:

```
var text = '{ "name":"testName", "age":"function () {return 30;}",
"city":"testCity"}';

var obj = JSON.parse(text);

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

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

JSON.stringify()

Stringify a JavaScript Object

When sending data to a web server, the data has to be a string.

Convert a JavaScript object into a string with JSON.stringify().

```
var obj = { "name":"testName", "age":30, "city":"testCity"};

var myJSON = JSON.stringify(obj);

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

Stringify a JavaScript Array

```
var arr = [ "Car", "Bike"];

var myJSON = JSON.stringify(arr);

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

Stringify Dates

```
var obj = { "name":"testName", "today":new Date(),
"city":"testCity"};

var myJSON = JSON.stringify(obj);

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

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:

```
var obj = { "name":"testName", "age":function () {return 30;},
"city":"testCity"};

var myJSON = JSON.stringify(obj);

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

convert your functions into strings before running the JSON.stringify() function.

```
var obj = { "name":"testName", "age":function () {return 30;},
"city":"testCity”};

obj.age = obj.age.toString();

var myJSON = JSON.stringify(obj);

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