



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

# 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

## 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":"testCity" };
var myJSON = JSON.stringify(myObj);
document.getElementById("demo").innerHTML = myJSON;
// 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": "testName", "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": [ "Car", "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]["name"] = "Mercedes";
myObj.cars[1]["models"] = ["Benz SLS AMG", "Benz CLA-Class"];
```

## Delete Array Items

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

## Parsing JSON

We received this text from a web server:

```
{ "name":"testName", "age":30, "city":"testCity" }
```

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

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

## JSON From the Server

Use the XMLHttpRequest to get data from the server:

```
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+"<br>" + myObj.age + "<br>";
```

```
        for (i in myObj.cars) {
```

```
            document.getElementById("demo").innerHTML += myObj.cars[i].name;
```

```
            document.getElementById("demo").innerHTML += "<li>" + myObj.cars[i].model + "</li>";
```

```
        }
```

```
    }
```

```
};
```

```
xmlhttp.open("GET", "json/json_demo.txt", true);
```

```
xmlhttp.send();
```

## Parsing Dates & Functions

### Array as JSON

Use the XMLHttpRequest to get data from the server:

```
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]+"<br>" + myArr[1];

    }

};

xmlhttp.open("GET", "json/json_demo_array.txt", true);

xmlhttp.send();
```

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

Convert a string into a date:

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

```
var myObj = JSON.parse(text);
```

```
myObj.birth = new Date(myObj.birth);
```

```
document.getElementById("demo").innerHTML = myObj.name
+ ", " + myObj.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 myObj = JSON.parse(text);
```

```
myObj.age = eval("(" + myObj.age + ")");
```

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



## 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 myObj = { "name":"testName", "age":30, "city":"testCity"};
```

```
var myJSON = JSON.stringify(myObj);
```

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

## Stringify a JavaScript Array

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

```
var myJSON = JSON.stringify(myArr);
```

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

## Stringify Dates

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

```
var myJSON = JSON.stringify(myObj);
```

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

## Stringify Functions

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

```
var myJSON = JSON.stringify(myObj);
```

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

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

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

```
myObj.age = myObj.age.toString();
```

```
var myJSON = JSON.stringify(myObj);
```

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

# HTML JSON

```
var txt, x;

xmlhttp = new XMLHttpRequest();

document.getElementById("head").innerHTML = "JSON HTML ";

xmlhttp.onreadystatechange = function() {

    if (this.readyState == 4 && this.status == 200) {

        myObj = JSON.parse(this.responseText);

        txt = "<table>";

        txt += "<tr><th>Color Name</th><th>Hex Color Code</th><th>RGB Color Code</th></tr>";

        for (x in myObj) {

            myObj[x].forEach(function(elem, index) {

                txt += "<tr><td style='color:" + elem.hexValue + "' class='colorName'>" + elem.colorName + "</td><td>" + elem.rgbValue + "</td><td>" + elem.hexValue + "</td></tr>";

            });

        }

        txt += "</table>"

        document.getElementById("demo").innerHTML = txt;

    }

};
```

## JSON with Ajax jQuery

```
$(document).ready(function () {  
  
    $('#get-demo-data').click(function () {  
  
        var demoData = $('#demo'), userData;  
  
        document.getElementById("head").innerHTML = "JSON with Ajax jQuery";  
  
        $.getJSON('json/json_ajax_jquery.json', function (data) {  
  
            userData = "<table id='user-table'>";  
  
            userData += "<tr><th>User ID</th><th>Name</th><th>User Name</th><th>Email</th><th>Address</th><th>Phone</th><th>Website</th><th>Company</th></tr>";  
  
            var users = data.users.map(function (user) {  
  
                userData += "<tr><td>" + user.id + "</td><td>" + user.name + "</td><td>" + user.username + "</td><td>" + user.email + "</td><td><b>Street:</b> " + user.address['street'] + "<br /><b>Suite:</b> " + user.address['suite'] + "<br /><b>City:</b> " + user.address['city'] + "<br /><b>Zip Code:</b> " + user.address['zipcode'] + "</td><td>" + user.phone + "</td><td><a href='"+user.website+"' target='_blank'>" + user.website + "</a></td><td><b>Name:</b> " + user.company['name'] + "<br /><b>Catch Phrase:</b> " + user.company['catchPhrase'] + "<br /><b>BS:</b> " + user.company['bs'] + "</td></tr>";  
  
            });  
  
            userData += "</table>"  
  
            document.getElementById("demo").innerHTML = userData;  
  
        });  
  
        demoData.text('Loading the JSON file.');
```

JSONP stands for JSON with Padding.

JSONP is a method for sending JSON data without worrying about cross-domain issues.

JSONP does not use the XMLHttpRequest object.

JSONP uses the <script> tag instead.

If you're developing a web-based application and are trying to load data from a domain which is not under your control, the chances are that you've seen the following message in your browser's console:

XMLHttpRequest cannot load http://external-domain/service. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://my-domain' is therefore not allowed access.

## A failing example

```
$(document).ready(function () {  
    $.getJSON(  
        "http://run.plnkr.co/plunks/  
v8xyYN64V4nqCshgjKms/data-1.json",  
        function(json) { console.log(json); }  
    );  
});
```

## A Possible Workaround

```
function jsonCallback(json){  
    console.log(json);  
}  
$.ajax({  
    url: "http://run.plnkr.co/plunks/v8xyYN64V4nqCshgjKms/  
data-2.json",  
    dataType: "jsonp"  
});
```



## JSONP specify a name for the callback function

```
function logResults(json){  
    console.log(json);  
}
```

```
$.ajax({  
    url: "https://api.github.com/users/jeresig",  
    dataType: "jsonp",  
    jsonpCallback: "logResults"  
});
```

(OR)

```
$.getJSON("https://api.github.com/users/jeresig?callback=?",function(json){  
    console.log(json);  
});
```

The **?** on the end of the URL tells jQuery that it's dealing with a JSONP request instead of JSON. jQuery then automatically registers the callback function which it calls when the request retruns.



# JSON Restructure

```
var response = [{
  "RegionName": "AMR",
  "SalesOrgCode": "1001",
  "SalesOrgDesc": "US",
  "Date": "2016-06-1"
}, {
  "RegionName": "AMR",
  "SalesOrgCode": "1002",
  "SalesOrgDesc": "Canada",
  "Date": "2016-06-2"
}, {
  "RegionName": "Europe",
  "SalesOrgCode": "2001  ",
  "SalesOrgDesc": "Netherlands",
  "Date": "2016-06-01"
}, {
  "RegionName": "Europe",
  "SalesOrgCode": "2002  ",
  "SalesOrgDesc": "UK",
  "Date": "2016-06-02"
}, {
  "RegionName": "APAC",
  "SalesOrgCode": "4001",
  "SalesOrgDesc": "New Zealand",
  "Date": "2016-06-01"
}, {
  "RegionName": "APAC",
  "SalesOrgCode": "4002",
  "SalesOrgDesc": "Singapore",
  "Date": "2016-06-02"
}];
```

```
var groups = {};
for(var g = 0; g < response.length; g++) {
  var item = response[g];
  if(item.RegionName != null) {
    if(!groups[item.RegionName]) {
      groups[item.RegionName] = [];
    }
    groups[item.RegionName].push({
      day: item.Date,
      orgcode: item.SalesOrgCode,
      orgdesc: item.SalesOrgDesc,
      sap: item.SAPCount,
      rover: item.ROVERCount
    });
  }
}
var datas = [];
for(var gs in groups) {
  if(Object.prototype.hasOwnProperty.call(groups, gs)) {
    var objGroup = {};
    objGroup[gs] = groups[gs];
    datas.push(objGroup);
  }
}
```

## JSON vs XML

	JSON	XML
<b>Stands For</b>	JSON : “JavaScript Object Notation”.	XML: “Extensible Markup Language”.
<b>Extended From</b>	JSON is extended from JavaScript.	XML is extended from SGML : “Standard Generalized Markup Language”.
<b>Purpose</b>	JSON is one type of text-based format or standard for interchanging data i.e. human readable. JSON is developed by “Douglas Crockford”.	XML is a Markup Language having format that contains set of rules for the encoding the documents which is readable for both human & machine. XML is developed by W3C :“World Wide Web Consortium “.
<b>Syntax</b>	JSON syntax is lighter than XML as JSON has serialized format of data having less redundancy. JSON does not contain start and end tags.	XML is not so lighter as JSON as having start and end tags and it takes more character than JSON to represent same data.
<b>Speed</b>	JSON is light – weighted in compare to XML as it has serialized format and so faster also.	XML is not so light weighted as JSON.
<b>Support of Data Type &amp; Array</b>	JSON supports datatype including integer and strings, JSON also supports array.	XML does not provide any data type so needs to be parsed into particular datatype. No direct support for array also.
<b>Comments</b>	JSON does not support Comments	XML supports comments.
<b>Parsing</b>	JSON uses only eval() for parsing i.e. for interpreting the JavaScript code & returns the result. It does not need any additional code for parsing.	XML needs XML Document Object Model (DOM) implementation & with that additional code for mapping text back to the JavaScript objects.
<b>Application</b>	For Web services, JSON is better.	For configuration, XML is better.

**Thank you**