

JSON and Domino

Bruce Lill

[Kalechi Designs](#)

Bruce Lill • February 26, 2008

2. Agenda

- Intro
- What is JSON
- How to use JSON
- JSON and Domino
- Demo of code
- Summary
- Questions

4. What Is JSON?

- It's **JavaScript Object Notation**
- Lightweight data-interchange format
- Simple format
- Easy for humans to read and write
- Easy for machines to parse and generate
- Programming language independent

NOTES

The format is readable but you can make objects that are very complicated to understand

5. Why JSON over XML

- Lighter and faster than XML as on-the-wire data format
- JSON objects are typed while XML data is typeless
 - JSON types: string, number, array, boolean, object
- XML data are all string
- Native data form for JavaScript code
 - Data is readily accessible as JSON objects in your JavaScript code vs. XML data needed to be parsed and assigned to variables.
 - Retrieving values is as easy as reading from an object property in your JavaScript code

6. The downside

- You need to know the JSON structure
 - There isn't a data definition
 - XML has the DTD or schema to describe the structure
 - XML has a namespace
 - Standard schemas available such as ODF
 - You can't easily navigate or convert the data
 - XML has XSLT's that let you navigate and convert the data.
 - Browsers can render XML
-

7. XML Sample

```
<?xml version="1.0" encoding="UTF-8"?>
<addressbook>
  <entry>
    <name>Bruce Lill</name>
    <address>
      <street>402 SW
Whiteridge</street>
      <city>Lees Summit, MO</city>
      <zip>64081</zip>
    </address>
    <phoneNumbers>
      <phone>816 246 4117</phone>
      <phone>816 588 7717</phone>
    </phoneNumbers>
  </entry>
</addressbook>
```

8. JSON - Sample

```
{"addressbook": {
  "name": "Bruce Lill",
  "address": {
    "street": "402 SW White Ridge",
    "city": "Lees Summit, MO",
    "zip": "64081"
  },
  "phoneNumbers": [
    "816 246 4117",
    "816 588 7717"
  ]
}
```

9. JSON Structure

- JSON can be either an Object or an Array
 - The Object is an unordered set of name/value pairs
 - An Object begins with { and end with }
 - A pair is a name followed by : and the value
 - The pairs are separated with a ,
 - {"name": "Bruce Lill" , "pet": "Binx"}
 - Value can be an Object, Array, string, number, boolean or Method
-

10. JSON Array

- The Array is an ordered set of enties
 - Array's begins with [and end with]
 - Each entry is separated by a ,
 - "myData": [1, false, "Lotus", [2, 3], {"count": 23}]
 - Entry can be an Object, Array, string, number, boolean or Method
-

11. JSON Data types

- Boolean - "active":true "active":false
- String - "address": "402 3rd Street"
- Positive integers - "quantity": 91
- Negative integers - "rating": -123
- Floats - "length":122.2344
- Scientific notation - "atoms per mole":-6.023e+23
- Null - "email":null

NOTES

In JSON, strings must be delimited by the double quote characters.
For a list of characters that are allowed and those that must be escaped, checkout the JSON web site.

12. JavaScript object sample

- JSON is a subset of the object notation of JavaScript
- Example: Javascript Object

```
var myJSONObject = {  
    "addressbook": {  
        "name": "Bruce Lill",  
        "address": {  
            "street": "402 SW White  
Ridge Drive",  
            "city": "Lees Summit, MO",  
            "zip": "64081"  
        },  
        "phoneNumbers": [  
            "816 246 4117",  
            "816 588 7717"  
        ]  
    }  
};
```

13. View data graphically

- JSON address object rendered by json2html

Object																													
Name	Value																												
addressbook	<table border="1"><thead><tr><th colspan="2">Object</th></tr><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>name</td><td>Bruce Lill</td></tr><tr><td>address</td><td><table border="1"><thead><tr><th colspan="2">Object</th></tr><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>street</td><td>402 SW White Ridge</td></tr><tr><td>city</td><td>Lees Summit, MO</td></tr><tr><td>zip</td><td>64081</td></tr></tbody></table></td></tr><tr><td>phoneNumbers</td><td><table border="1"><thead><tr><th colspan="2">Array</th></tr><tr><th>Index</th><th>Value</th></tr></thead><tbody><tr><td>0</td><td>816 246 4117</td></tr><tr><td>1</td><td>816 588 7717</td></tr></tbody></table></td></tr></tbody></table>	Object		Name	Value	name	Bruce Lill	address	<table border="1"><thead><tr><th colspan="2">Object</th></tr><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>street</td><td>402 SW White Ridge</td></tr><tr><td>city</td><td>Lees Summit, MO</td></tr><tr><td>zip</td><td>64081</td></tr></tbody></table>	Object		Name	Value	street	402 SW White Ridge	city	Lees Summit, MO	zip	64081	phoneNumbers	<table border="1"><thead><tr><th colspan="2">Array</th></tr><tr><th>Index</th><th>Value</th></tr></thead><tbody><tr><td>0</td><td>816 246 4117</td></tr><tr><td>1</td><td>816 588 7717</td></tr></tbody></table>	Array		Index	Value	0	816 246 4117	1	816 588 7717
Object																													
Name	Value																												
name	Bruce Lill																												
address	<table border="1"><thead><tr><th colspan="2">Object</th></tr><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>street</td><td>402 SW White Ridge</td></tr><tr><td>city</td><td>Lees Summit, MO</td></tr><tr><td>zip</td><td>64081</td></tr></tbody></table>	Object		Name	Value	street	402 SW White Ridge	city	Lees Summit, MO	zip	64081																		
Object																													
Name	Value																												
street	402 SW White Ridge																												
city	Lees Summit, MO																												
zip	64081																												
phoneNumbers	<table border="1"><thead><tr><th colspan="2">Array</th></tr><tr><th>Index</th><th>Value</th></tr></thead><tbody><tr><td>0</td><td>816 246 4117</td></tr><tr><td>1</td><td>816 588 7717</td></tr></tbody></table>	Array		Index	Value	0	816 246 4117	1	816 588 7717																				
Array																													
Index	Value																												
0	816 246 4117																												
1	816 588 7717																												

14. Using the JSON object

- Once you have JSON object, you can use dot notation to access its properties
 - `var name = JSONdata.name;`
 - `var phone = JSONdata.phone[0];`
 - `JSONdata.Method()`
 - Or using array notation
 - `var name = JSONdata["name"]`
 - `var phone = JSONdata["phone"][0]`
 - `JSONdata["Method()"]`
-

15. JSON data is received as a string

- Usually it's received from an XMLHttpRequest
 - The string can be sent as text or json format
 - `Print |Content-type: application/json|`
 - `Print |Content-type: text/plain|`
 - To convert JSON text into an JS object, use `eval()`
 - `eval()` invokes the JavaScript compiler
 - Since JSON is a proper subset of JavaScript, the compiler will correctly parse the text and produce an object structure
 - `var data = eval('(' + JSONtext + ')');`
-

16. Security & JSON Parser

- `eval()` can compile and execute any JavaScript program, so there can be security issues (cross-site scripting)
 - When security is a concern then use a JSON parser
 - The JSON parser will only recognize JSON text and so is much safer
 - `var JSONdata = JSON.parse(txt);`
 - Or use one of the libraries that support security such as Prototype.
-

17. Getting JSON from Domino

- Views
 - Pull data from any column
 - Fast
 - Don't forget to add `&count=0`
 - Page
 - Use Computed-Text, embedded views, dblookup or dbcolumn
 - Can pre-process data
 - Agents
 - Data from one or more databases
 - Can filter & convert data
 - Can be LotusScript or Java
-

18. Domino view JSON

- `notesini.nsf/name?readviewentries&count=0&outputformat=json`

```
{"@toplevelentries": "2471",
"viewentry": [
  {"@position": "1",
"@unid": "6FD5F806C8D5FDD5C1256E9E002C71B0",
"@noteid": "17F6",
"@siblings": "2471",
"entrydata": [
  {"@columnnumber": "0",
"@name": "$38",
"text": {"0": "[<a
href=\"\/notesini\/name\/ActionPaneEnabled?open&vw=name\">Acti
onPaneEnabled<\/a>"]}},
  {"@columnnumber": "1",
"@name": "$37",
"number": { "0": "0"}},
  {"@columnnumber": "2",
"@name": "$36",
"text": {"0": "0 / 1"}}
]
}
]
```

19. View some code

- Sample Javascript:

```
var txt = xhr.responseText;
var data = JSON.parse(txt);
//json.js var data = txt.parseJSON();
//plain var data=eval("(" + text + ")");
var top = parseInt(data["@toplevelentries"]);
```

20. Summary

- Get Firefox and web developer, Firebug, & Tamper Data extensions
- Pick your ajax library
- Pick the method for getting data from Domino
- Define your data structure
- Put it together

21. Resources

- The home of JSON - www.json.org
- The home of JSONRequest - www.json.com
- JSON to html - json.budgetwebdesign.org
- JSON Guide - www.1060research-server-1.co.uk/docs/3.1.0/book/discovered/doc_mod_json_guide.html
- Dojo - dojotoolkit.org
- Prototype - www.prototypejs.org
- Victor's JSON Class Library - download.snapps.com

22. Questions?