

# Introduction to JSON (JavaScript Object Notation)

**Sang Shin**  
**Java Technology Architect**  
**Sun Microsystems, Inc.**  
**[sang.shin@sun.com](mailto:sang.shin@sun.com)**  
**[www.javapassion.com](http://www.javapassion.com)**

# Disclaimer & Acknowledgments

- Even though Sang Shin is a full-time employee of Sun Microsystems, the contents here are created as his own personal endeavor and thus does not reflect any official stance of Sun Microsystems

# Topics

- What is JSON?
- JSON Data Structure
  - > JSON Object
  - > JSON text
- JSON and Java Technology
- How to send and receive JSON data at both client and server sides
- Resources

# What is JSON?

# What is JSON?

- Lightweight data-interchange format
  - > Compared to XML
- Simple format
  - > Easy for humans to read and write
  - > Easy for machines to parse and generate
- JSON is a text format
  - > Programming language independent
  - > Uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python

# 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,
  - > 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 through tedious DOM APIs
  - > Retrieving values is as easy as reading from an object property in your JavaScript code

# Where is JSON Used?

- Represent configuration information
- Implement communication protocols

# JSON Object



# JSON Structures

- A collection of name/value pairs
  - > In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array
- An ordered list of values
  - > In most languages, this is realized as an array, vector, list, or sequence
- These are universal data structures supported by most modern programming languages

# JSON Object Notation

- A JSON object is an unordered set of name/value pairs
- A JSON object begins with { (left brace) and ends with } (right brace)
- Each name is followed by : (colon) and the name/value pairs are separated by , (comma)

# JSON and JavaScript

- JSON is a subset of the object literal notation of JavaScript
  - > JSON can be used in the JavaScript language with no muss or fuss

# Example: JSON Object

```
var myJSONObject = {"bindings": [  
    {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},  
    {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},  
    {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}  
    ]  
};
```

- In this example, a JSON JavaScript object is created containing a single member "bindings", which contains an array containing three objects, each containing "ircEvent", "method", and "regex" members
- Members can be retrieved using dot or subscript operators

```
myJSONObject.bindings[0].method // "newURI"
```

# Text to Object Conversion in JavaScript code

```
var myObject = eval('(' + myJSONtext + ');');
```

- To convert a JSON text into an JSON object, use the `eval()` function
  - > `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

# Security & JSON Parser

```
// Include http://www.json.org/json.js  
var myObject = myJSONtext.parseJSON();
```

- `eval()` can compile and execute any JavaScript program, so there can be security issues (cross-site scripting)
  - > Use `eval()` when the source can be trusted
- When security is a concern - the source cannot be trusted -, it is better to use a JSON parser
  - > A JSON parser will only recognize JSON text and so is much safer

# Object to Text Conversion

```
var myJSONText = myObject.toJSONString();
```

- You can convert JSON object into JSON text
- JSON does not support cyclic data structure
  - > Do not give cyclical structures to the JSON stringifier

# JSON in Java



# JSON Tools for Java Developer

- Parser
  - > Parse JSON text files and convert these to a Java model
- Renderer
  - > Render a Java representation into text
- Serializer
  - > Serialize plain POJO clusters to a JSON representation
- Validator
  - > Validate the contents of a JSON file using a JSON schema

# JSONObject Java Class

- A `JSONObject` is an unordered collection of name/value pairs
- The `put` methods adds a name/value pair to an object
- The texts produced by the `toString` methods strictly conform to the JSON syntax rules

```
myString = new JSONObject().put("JSON", "Hello, World!").  
    toString();  
// myString is {"JSON": "Hello, World"}
```

# How to Send & Receive JSON Data at Both Client and Server Side

# How to Generate/Send JSON Data at the Server Side

- Create `JSONObject` Java object
- Add name/value pairs using `put` method
- Convert it to `String` type using `toString` method and send it to the client with content-type as “text/xml” or “text/plain”

```
myString = new JSONObject().put("JSON", "Hello, World!").  
    toString();  
// myString is {"JSON": "Hello, World"}
```

# How to Receive JSON Data at the Client Side

- JSON data is received as a string
- Calling `eval()` will generate JSON object in JavaScript code
  - > `var JSONdata = eval(req.responseText);`
- Once you have JSON object, you can use `.` notation to access its properties
  - > `var name = JSONdata.name;`
  - > `var address = JSONdata.addresses[3];`
  - > `var streetname = JSONdata.addresses[3].street;`

# How to Generate/Send JSON Data at the Client Side

- Create JSON JavaScript object
- Use “POST” HTTP method in the `open` method of the XMLHttpRequest object
- Pass JSON JavaScript object in the `send` method of XMLHttpRequest object

```
var carAsJSON = JSON.stringify(car);  
var url = "JSONExample?timeStamp=" + new Date().getTime();  
createXMLHttpRequest();  
xmlHttp.open("POST", url, true);  
xmlHttp.onreadystatechange = handleStateChange;  
xmlHttp.setRequestHeader("Content-Type",  
                           "application/x-www-form-urlencoded");  
xmlHttp.send(carAsJSON);
```

# How to Receive JSON Data at the Server Side

- Read the JSON data as a String type
- Create **JSONObject** Java object from the string

```
String json = readJSONStringFromRequestBody(request);
//Use the JSON-Java binding library to create a JSON object in Java
JSONObject jsonObject = null;
try {
    jsonObject = new JSONObject(json);
}
catch(ParseException pe) {
}
```

# Resources



# JSON Resources

- Introducing JSON
  - > <http://www.json.org/>
- JSON in JavaScript
  - > <http://www.json.org/js.html>
- JSON in Java
  - > <http://www.json.org/java/index.html>

# Introduction to JSON (JavaScript Object Notation)

**Sang Shin**  
Java Technology Architect  
Sun Microsystems, Inc.  
[sang.shin@sun.com](mailto:sang.shin@sun.com)  
[www.javapassion.com](http://www.javapassion.com)