JSON-RPC

JavaScript Object Notation
Remote Procedure Calls

A presentation by Derrell Lipman for Internet & Web Systems II, Spring, 2010
Outline

- Background – why are we discussing this?
- What is JavaScript Object Notation?
- What is a Remote Procedure Call?
- Using JSON-RPC
- Examples
- Related Technologies
Background Evolution

- Static web pages
- Form Submission
- Client-side form validation
- XMLHttpRequest

- and then...
Announcing JSON-RPC:

- JavaScript Object Notation Remote Procedure Calls
JSON-RPC

Constituent Parts

- JSON – JavaScript Object Notation
- RPC – Remote Procedure Calls
JavaScript Object Notation

- Primitive types: null, boolean, string, number
- Structured types: object, array
Primitive Types: Number, String

- Examples: 42, 3.33, “Hello world”
Structured Type: Array

Example: [ 23, 42, “abc”, 3.1415926 ]
Structured Type: Object

- Example: `{ "key" : 23, "format" : "ascii" }`
Remote Procedure Calls
Putting It All Together

- JSON-RPC is a simple, light-weight remote procedure call protocol
- Supports: Request, Response, Notification
A function call is issued by passing a Request Object:

- "jsonrpc" : "2.0"
- "method" : <string>
- "params" : <structured value>
- "id" : <string>, <number>, or null
Parameter Encoding

- **By position**
  - Passed as an Array of values in expected order

- **By name**
  - Passed as an Object
  - Each member name must be a parameter name
  - Value corresponding to that name is passed as parameter value
Example Request

{  
  "jsonrpc": "2.0",
  "method": "add",
  "params": [ 23, 42 ],
  "id": 1
}

Equivalently, since it need not be easily human readable:

{"jsonrpc":"2.0","method":"add","params":[23,42],"id":1}
The return value of a function call is provided in a Response Object:

- "jsonrpc": "2.0"
- "result": <determined by called function>
  - Provided only upon success
- "error": <Error object>
  - Provided only upon error
- "id": <string>, <number>
  - Correspondence between Request and Response
Example:
RPC with positional parameters

Request
{
    "jsonrpc" : "2.0",
    "id" : 1,
    "method" : "add",
    "params" : [42, 23]
}

Response
{
    "jsonrpc" : "2.0",
    "id" : 1,
    "result" : 65
}
Example: RPC with named parameters

Request
{
   "jsonrpc" : "2.0",
   "id" : 2,
   "method" : "subtract",
   "params" : { "subtrahend" : 23, "minuend" : 42 }
}

Response
{
   "jsonrpc" : "2.0",
   "id" : 2,
   "result" : -19
}
Error Object

- Object provided in the “error” field of a response when an error has occurred
  - “code” : <Integer>
  - “message” : <short, descriptive textual explanation>
  - “data” : <primitive or structured data providing detail>
Example:
Invalid JSON

Request
{
    "jsonrpc" : "2.0",
    "id" : 3,
    "method" : "update",
    "params" : "foo", "bar", 10 ]
}

Response
{
    "jsonrpc" : "2.0",
    "error" : { "code" : -32700, "message" : "Parse error" },
    "id" : 3
}
Example:

Non-existent method

Request
{
   "jsonrpc" : "2.0",
   "id" : 4,
   "method" : "foobar",
   "params" : [ 1, 5, 10 ]
}

Response
{
   "jsonrpc" : "2.0",
   "id" : 4,
   "error" : { "code" : -32601, "message" : "Method not found" }
}
Notification (special Request)

- One-way (client → server) message
- Request with “id” field set to null (or missing)
- No Response sent to this Request
Example: Notification

Request
{
    "jsonrpc": "2.0",
    "method": "keepalive",
    "params": { "seconds": 60 }
}

Batch Requests

- Array of Request Objects
- Server may process Requests concurrently, or serially in any order
- Response is an array of Response Objects in any order
- Client matches Response to appropriate Request using “id” value
- Notifications in Request Array have no Response Object in Response Array
Related Technology:

JSONP

- JSONP
  - Adds a reply expression or function call
  - Helps for cross-domain requests
  - Typically uses `<script>` tags for RPC
Example: JSONP

Request
{"jsonrpc":"2.0","id":4,"request":"add","params":[2,3]}
jsonp=requestCallback

http://some.site.com/?rpc=%22jsonrpc%22%3A%222.0%22%2C%22id%22%3A%4%2C%22request%22%3A%22add%22%2C%22params%22%3A%5B2%2C3%5D%7D&jsonp=requestCallback

Response
requestCallback({"jsonrpc":"2.0","id":4,"response":5})
<?xml version="1.0"?>
<methodCall>
   <methodName>
      subtract
   </methodName>
   <params>
      <param>
         <value>
            <int>23</int>
         </value>
      </param>
      <param>
         <value>
            <int>42</int>
         </value>
      </param>
   </params>
</methodCall>
Related Technology: XML-RPC comparison to JSON-RPC

```xml
<?xml version="1.0"?>
<methodCall>
    <methodName>
        subtract
    </methodName>
    <params>
        <param>
            <value>
                <int>23</int>
            </value>
        </param>
        <param>
            <value>
                <int>42</int>
            </value>
        </param>
    </params>
</methodCall>
```

```json
{
    "jsonrpc": "2.0",
    "id": 2,
    "method": "subtract",
    "params": [23, 42]
}
```
JSON-RPC specification and discussions are at http://groups.google.com/group/json-rpc

JSON specification is RFC-4627 available at http://www.faqs.org/rfcs/rfc4627.html

Also see http://json.org

JSONP is from http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/

Google recommends a JSONP extension called JSONPP http://sites.google.com/a/s3db.org/s3db/documentation/mis/json-jsonp-jsonpp

XML-RPC is specified at http://www.xmlrpc.com/spec