JSON

Dr. Kanda Runapongsa Saikaew
Computer Engineering Department
http://twitter.com/krunapon

Agenda

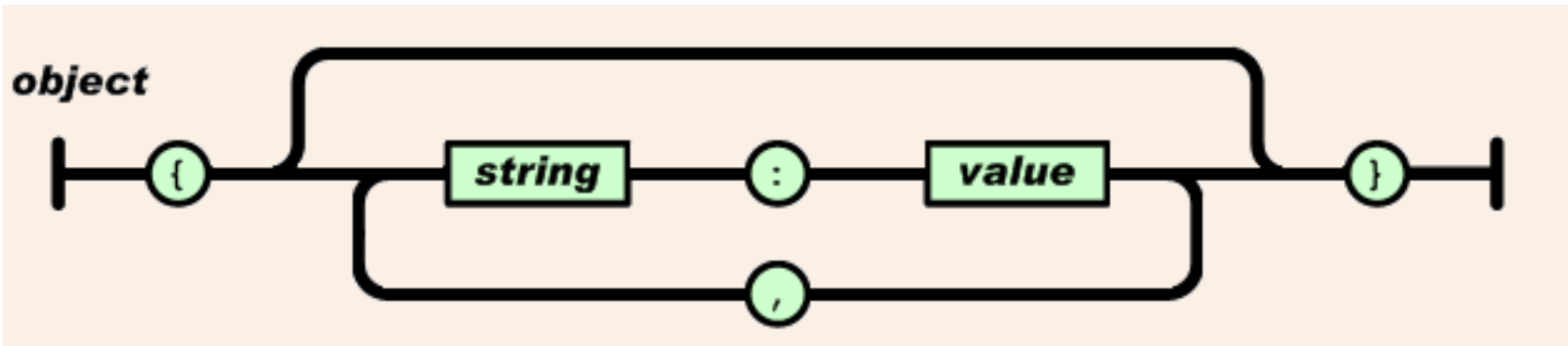
- What is JSON?
- JSON vs. XML
- JSON Tutorial
- JSON and AJAX
- Using JSON with Yahoo! Web Services
- Using JSON with Google Data Protocol

What is JSON?

- **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.
- JSON is a text format that is completely language independent
- JSON is built on two 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.

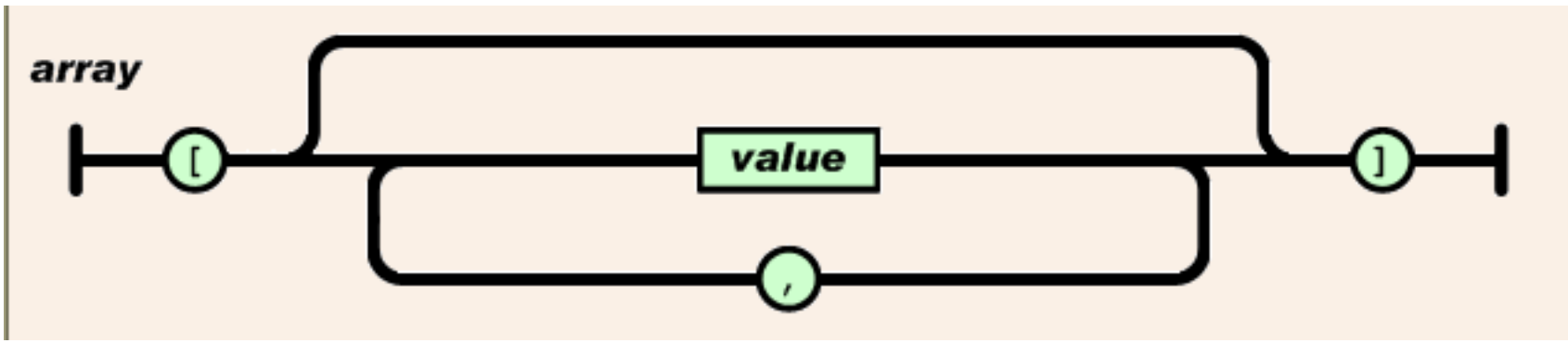
JSON Forms: An object

- An *object* is an unordered set of name/value pairs
- An 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)



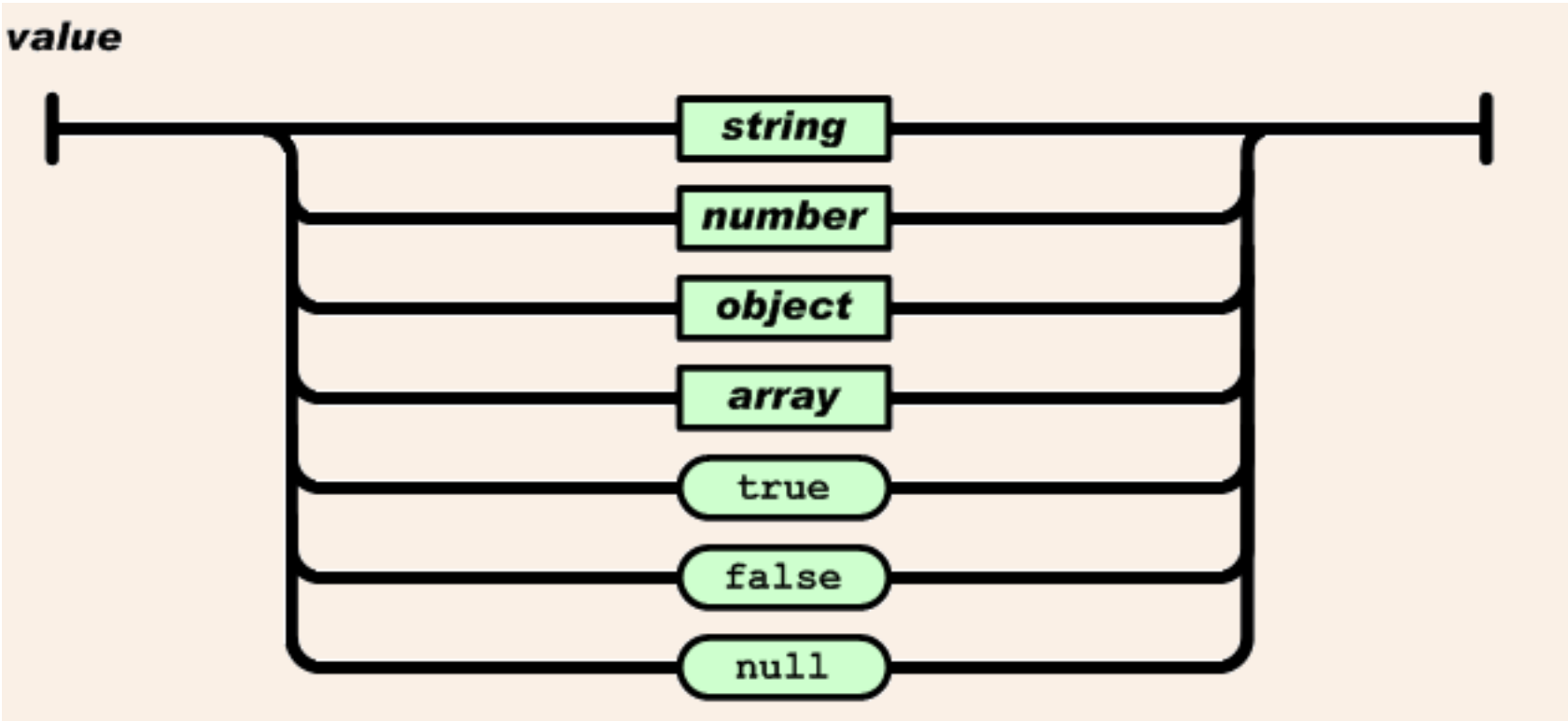
JSON Forms: An Array

- An *array* is an ordered collection of values
- An array begins with [(left bracket) and ends with] (right bracket)
- Values are separated by , (comma).



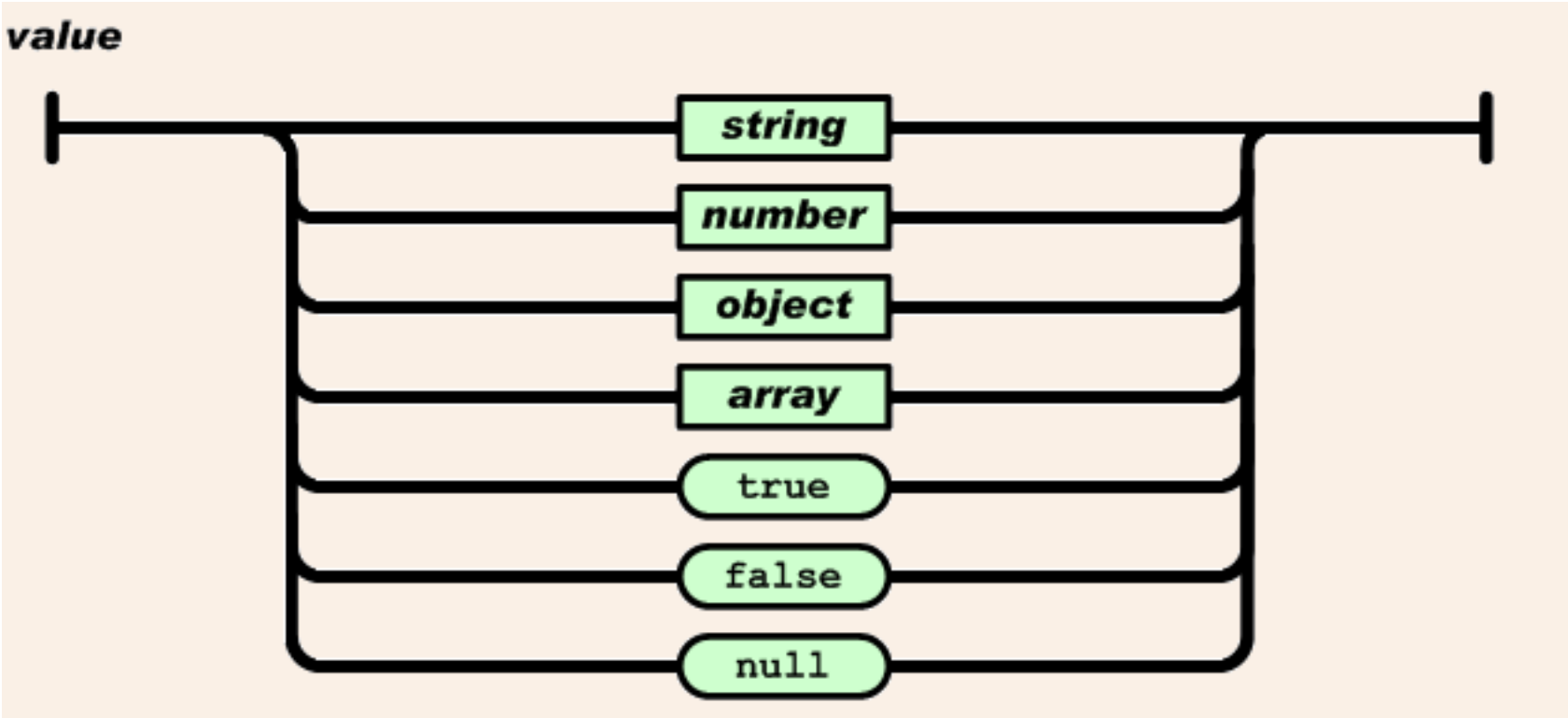
JSON Forms: A Value

- A *value* can be a *string* in double quotes, or a *number*, or true or false or null, or an *object* or an *array*
- These structures can be nested.



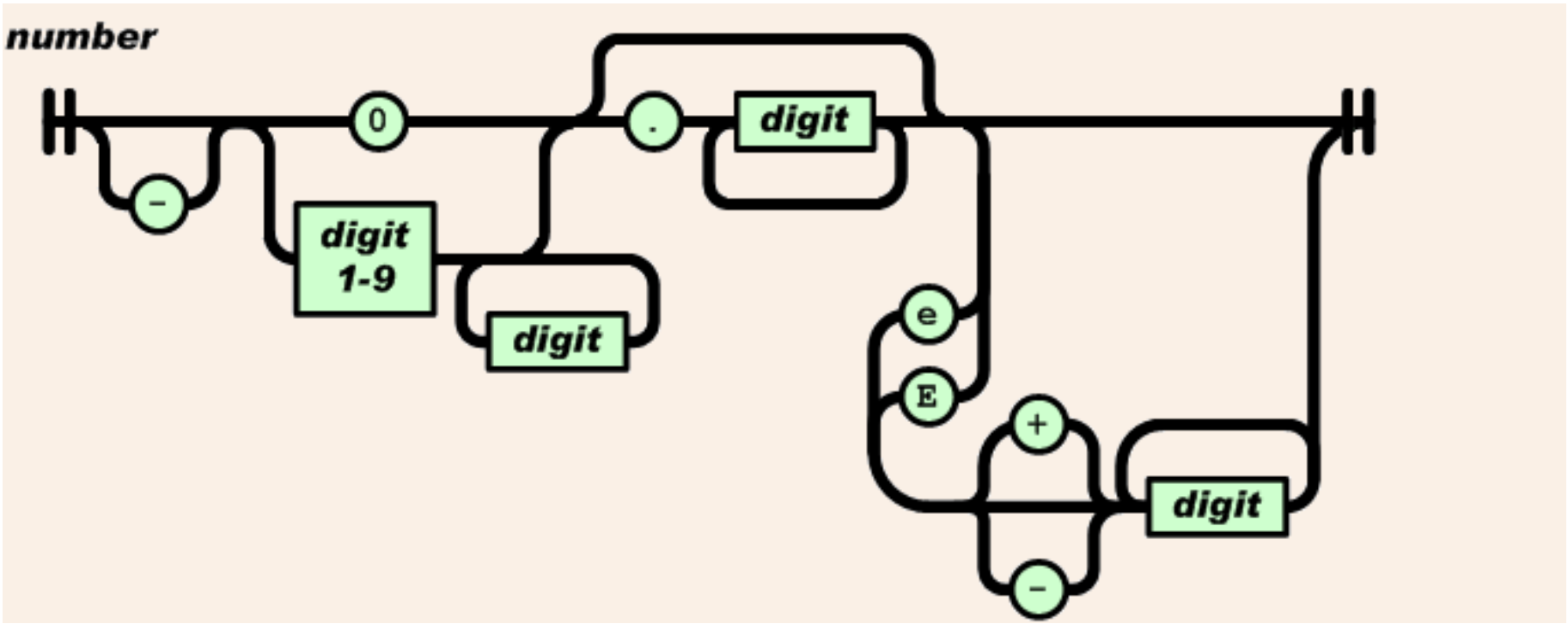
JSON Forms: A String

- A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes
- A character is represented as a single character string
- A string is very much like a C or Java string



JSON Forms: A Number

- A *number* is very much like a C or Java number, except that the octal and hexadecimal formats are not used



JSON Sample

{"skillz": { "web":[{"name": "html", "years": "5" }, {"name": "css", "years": "3" }], "database":[{"name": "sql", "years": "7" }] }}

Squiggles, Squares, Colons, and Commas

1. Squiggly brackets act as 'containers'
    - {...}
2. Square brackets holds arrays
    - [...]
3. Names and values are separated by a colon
    - name:value
4. Array elements are separated by commas
    - [member1, member2]

JSON is like XML because

1. They are both 'self-describing' meaning that values are named, and thus 'human readable'
2. Both are hierarchical. (i.e. You can have values within values.)
3. Both can be parsed and used by lots of programming languages
4. Both can be passed around using AJAX (i.e. httpWebRequest)

JSON is Unlike XML because

1. XML uses angle brackets, with a tag name at the start and end of an element: JSON uses squiggly brackets with the name only at the beginning of the element.

2. JSON is less verbose so it's definitely quicker for humans to write, and probably quicker for us to read.

3. JSON can be parsed trivially using the eval() procedure in JavaScript

4. JSON includes arrays {where each element doesn't have a name of its own}

5. In XML you can use any name you want for an element, in JSON you can't use reserved words from javascript

JSON in JavaScript

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

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

- In this example, an object is created containing a single member "bindings", which contains an array containing three objects, each containing "ircEvent", "method", and" regex" members.

Accessing JSON in JavaScript

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

- Members can be retrieved using dot or subscript operators.

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

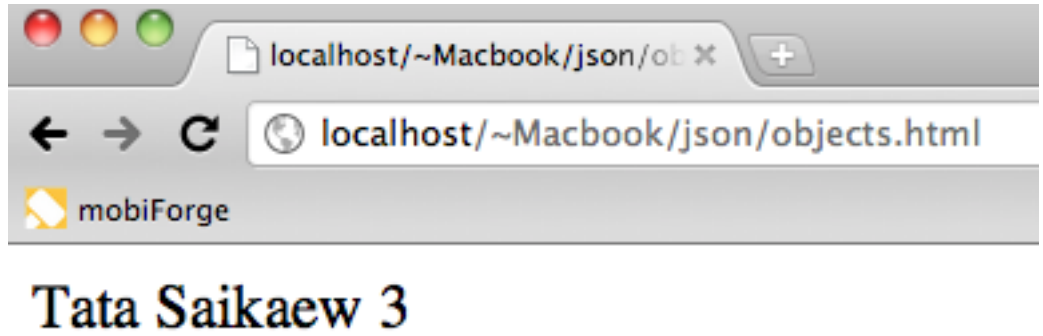
- To convert a JSON text into an object, you can use the eval() function. eval() invokes the JavaScript compiler
- The text must be wrapped in parens to avoid tripping on an ambiguity in JavaScript's syntax.

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

Safely Parsing JSON

- The eval function is very fast. However, it can compile and execute any JavaScript program, so there can be security issues
- The use of eval is indicated when the source is trusted and competent. It is much safer to use a JSON parser
- Modern browsers, such as Firefox 3.5 and Internet Explorer 8, include special features for parsing JSON
- As native browser support is more efficient and secure thaneval(), it is expected that native JSON support will be included in the next ECMAScript standard

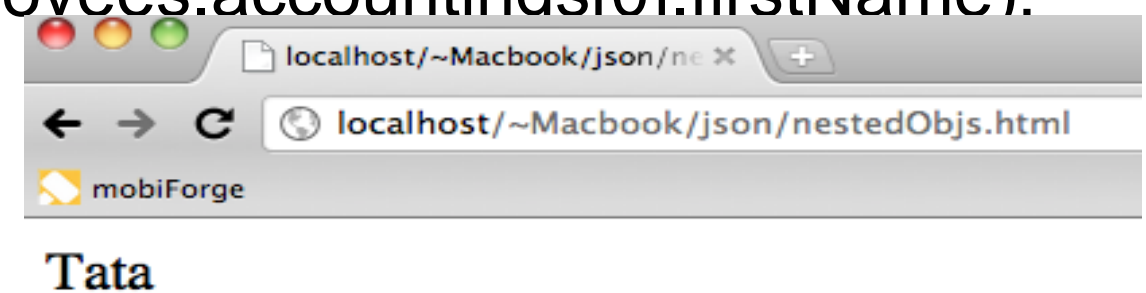
Objects as Data




```
<html>
<body>
<script type="text/javascript">
var myFirstJSON =  {"firstName" : "Tata",
                    "lastName"  : "Saikaew",
                    "age"  : 3};
document.writeln(myFirstJSON.firstName);
document.writeln(myFirstJSON.lastName);
document.writeln(myFirstJSON.age);
</script>
</body>
</html>
```

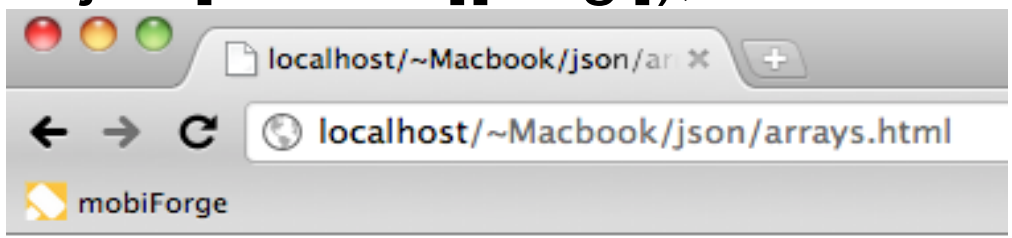
Nested Objects

```
<html>
<body>
<script type="text/javascript">
var employees = { "accountings" : [ // accounting is an array in
employees
{"firstName"  : "Tata", "lastName" : "Saikaew", "age" : 30},
{"firstName" : "Teetee", "lastName" : "Wongkaew", "age" : 26} ],
"sales" : [ // Sales is another array in employees
{ "firstName" : "Manee", "lastName" : "Jaidee", "age" : 28},
{ "firstName" : "Mana", "lastName" : "Deejai", "age" : 32}]
}; // End employees
document.write(employees.accountings[0].firstName);
</script>
</body>
</html>
```



Simulating An Associated Array

```
<html>
<body>
<script type="text/javascript">
  var myObject = {'color' : 'blue',
                  'animal' : {'dog' : 'friendly' }
                };
  document.writeln(myObject.animal.dog);
  document.writeln(myObject['color']);
  document.writeln(myObject['animal']['dog']);
</script>
</body>
</html>
```



XML File

```
<wclass>
<!--My students who took web programming class with me-->
<student id="1">
<name>Linda Jones</name>
<legacySkill>Access, VB5.0</legacySkill>
</student>
<student id="2">
<name>Adam Davidson</name>
<legacySkill>Cobol, MainFrame</legacySkill>
</student>
<student id="3">
<name>Charles Boyer</name>
<legacySkill>HTML, Photoshop</legacySkill>
</student>
</wclass>
```

AJAX and XML Code (1/4)

```
<html>
   <head>
      <title>AJAX and XML</title>
      <script language="javascript">
         var xhr = false;
         function GetXmlHttpObject(){
            var xmlHttp=null;
            try {
               // Firefox, Opera 8.0+, Safari
               xmlHttp=new XMLHttpRequest();
            } catch (e) {
               // Internet Explorer
               try {
                  xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
               } catch (e) {
                  xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
               }
            }
            return xmlHttp;
         }
```

AJAX and XML Code (2/4)

```
function getPage (url) {
        xhr = GetXmlHttpObject();
        if (!xhr) {
            alert ('XMLHttp failed to instantiate');
            return false;
        }
        xhr.onreadystatechange = statusCheck;
        xhr.open ('GET', url, true);
        xhr.send (null);
    }
```

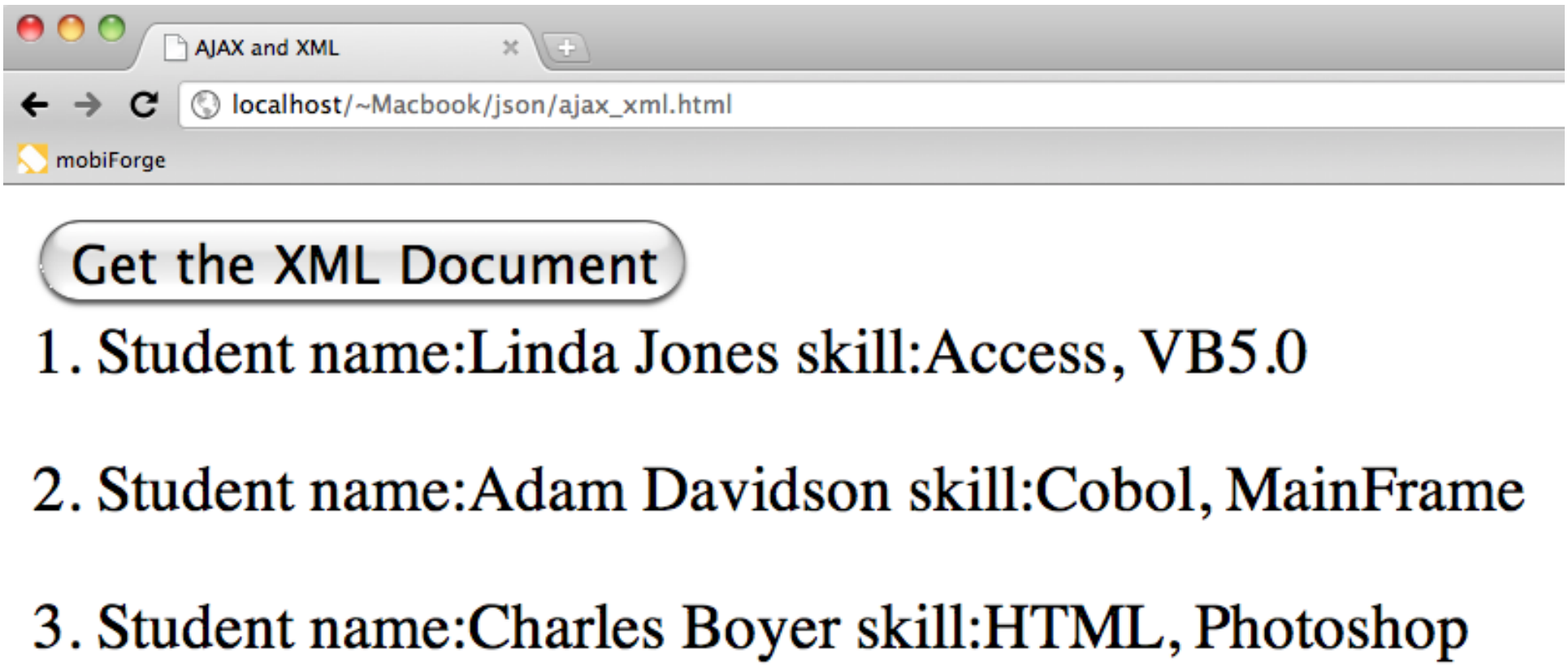
AJAX and XML Code (3/4)

```
 function statusCheck() {
            if (xhr.readyState == 4) {
                  if (xhr.status == 200) {
                     var nodes=xhr.responseXML.getElementsByTagName
("student");
                     j = 1;
                     for (i=0; i<nodes.length; i++) {
                        var student = nodes.item(i);
                        document.getElementById(j++).innerHTML= (i+1) + ".
Student name:" +
                        student.firstChild.nextSibling.firstChild.nodeValue + "  skill:"
+
                        student.lastChild.previousSibling.firstChild.nodeValue;
                     }
                  } else {
                     alert ('The request could not be fulfilled.');
                  }
            }
         }
   </script>
```

AJAX and XML Code (4/4)

```
</head>
    <body>
        <input id=button2 onclick=
                "getPage ('http:
///localhost/~Macbook/json/webstudents.xml')" type=button
                value="Get the XML Document" name=button2>
        <br/>
        <div id=1></div></p>
        <div id=2></div></p>
        <div id=3></div></p>
    </body>
</html>
```

AJAX and XML Code Result



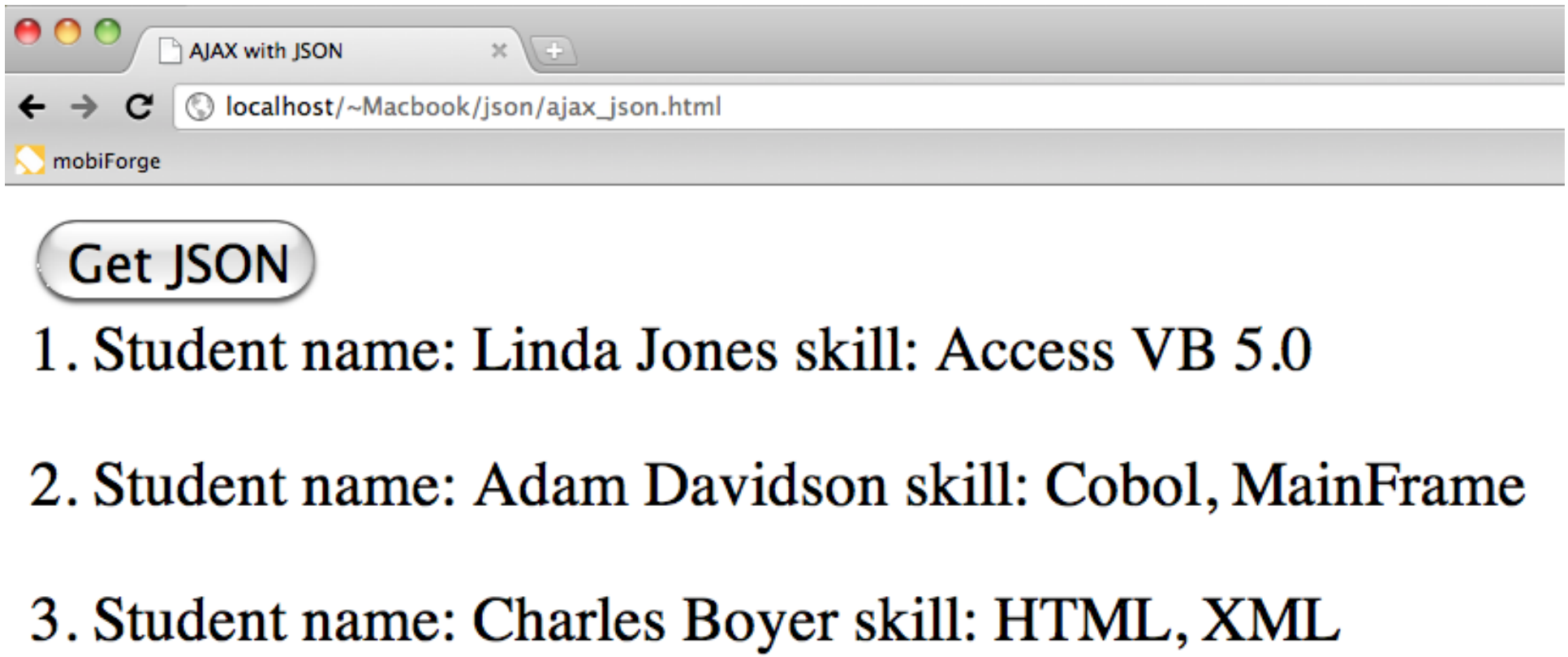
JSON File


```
{"wclass": [
 {"student": {"@attributes" : { "id" : "1" }, "name": "Linda
Jones",              "legacySkill": "Access VB 5.0"
          }},
 {"student": {"@attributes" : { "id": "2"  },"name": "Adam
Davidson",
              "legacySkill": "Cobol, MainFrame"
          }},
 {"student": {"@attributes" : { "id": "3" },"name": "Charles Boyer",
               "legacySkill": "HTML, XML"
          }
      }
   ]
}
```

JSON and JQuery

```
<script type="text/javascript" src="jquery-1.5.js"></script>
<script type="text/javascript">
    ...
        function statusCheck() {
            if (xhr.readyState == 4) {
                if (xhr.status == 200) {
                    var jsonT = xhr.responseText;
                    var obj = jQuery.parseJSON(jsonT);
                    var wclass = obj.wclass;
                    j = 1;
                    for (i = 0; i < wclass.length; i++) {
                            document.getElementById(j++).
innerHTML=
                            (i+1) + ". Student name: " +     wclass[i].
student.name + " skill: " + wclass[i].student.legacySkill +
"<br/>";
                    }}}
```

JSON and JQuery Result



Agenda

- What is JSON?
- JSON vs. XML
- JSON Tutorial
- JSON and AJAX
- **Using JSON with Yahoo! Web Services**
- Using JSON with Google Data Protocol

How To Request JSON Output

Many Yahoo! Web Services that support JSON use common parameters for generating and working with JSON output.

- Get JSON Output with output=json
- Use Callbacks with callback=function

USING OUTPUT=JSON

By default the Yahoo! Web Services return output in XML format. To get output in JSON format, use theoutput=json parameter in the request:
http://search.yahooapis.com/ImageSearchService/V1/imageSearch?appid=YahooDemo&query=Madonna&output=json

Using callback=function

- The callback parameter (callback=*function*) wraps the JSON output text in parentheses and a function name of your choosing
- For example:

http://search.yahooapis.com/ImageSearchService/V1/imageSearch?appid=YahooDemo&query=Madonna&output=json&callback=ws_results

- Callback function names may only use upper and lowercase alphabetic characters (A-Z, a-z), numbers (0-9), the period (.), the underscore (_), and brackets ([and]). Brackets must be URL-encoded;

Callbacks

- Results wrapped in callbacks look like this:

ws_results(*...json output...*);

- Because JSON output is native JavaScript, you do not have to parse or evaluate the returned object in your callback function
- You can immediately access the elements inside it, just as if you had passed an object reference to your ws_result function.

Callbacks as Objects

- In addition to a simple function name, callbacks can also use JavaScript object or array references, like this:

callback=ws_results.obj

... or this:

callback=ws_results.obj.array[4]

- Brackets requested in callbacks must be URL encoded, like this:

http://search.yahooapis.com/ImageSearchService/V1/imageSearch?appid=YahooDemo&query=Madonna&output=json&callback=ws_results.obj.array%5B4%5D

Callbacks and JSON inside <script>

- Callbacks are particularly useful for use with web service requests in client-side JavaScript
- Normally web service requests using the XMLHttpRequest object run afoul of browser security restrictions that prevent files from being loaded across domains
- Using JSON and callbacks, you can place the Yahoo! Web Service request inside a <script> tag, and operate on the results with a function elsewhere in the JavaScript code on the page.

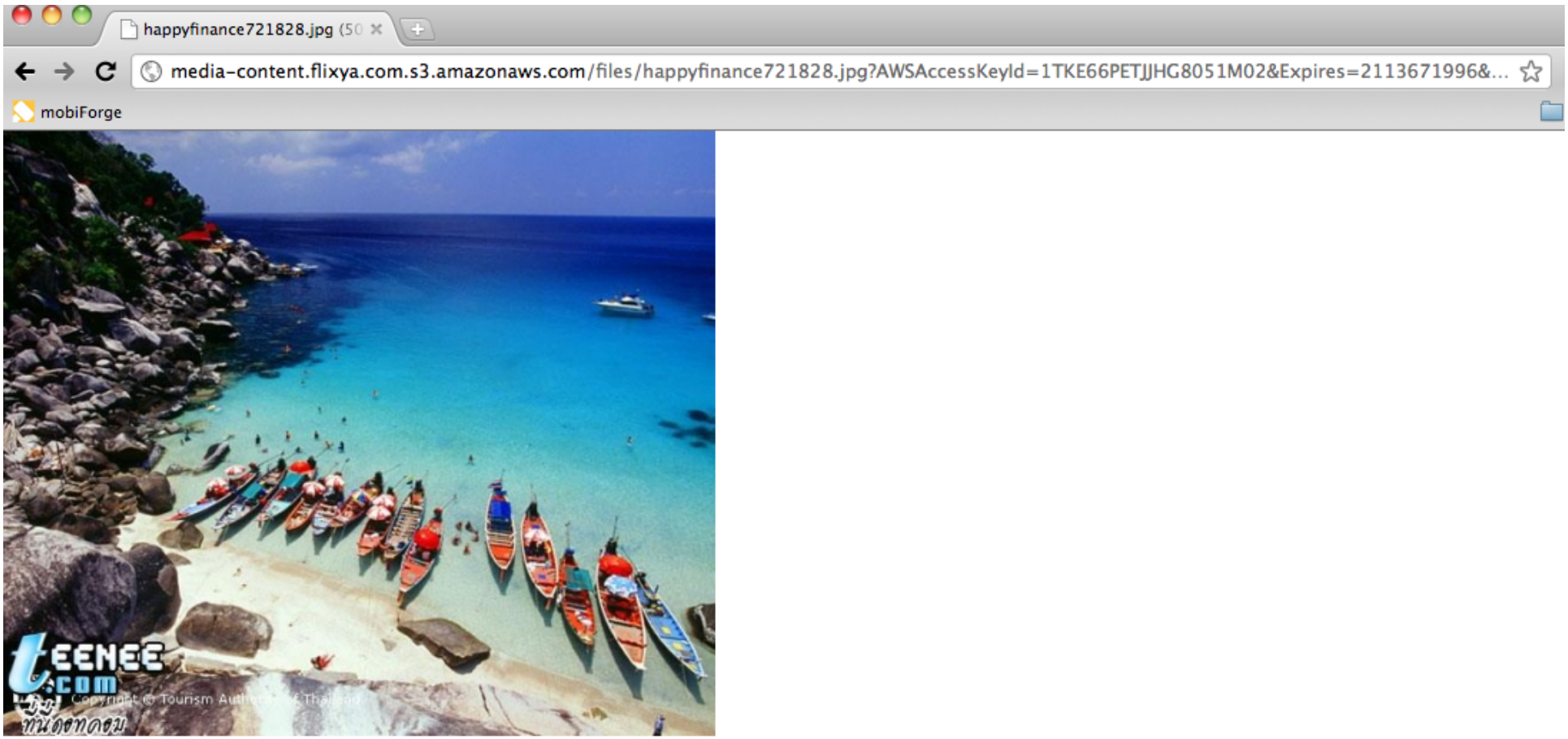
Sample Script to Call Yahoo API

```
<html> <head> <title>How Many Pictures Of Thailand Do We Have?</title>
</head></body> <script type="text/javascript"> function ws_results(obj) { document.write
("Total results:" + obj.ResultSet.totalResultsAvailable + "<br/>"); var results = obj.
ResultSet.Result; var numResults = results.length; for (i = 0; i < numResults; i++) {
document.write((i+1) + ":" + results[i].Title); var url = results[i].ClickUrl; document.write(":<a
href='" + url + "'>" + url + "</a><br/>"); } } </script></html>
```

Call Yahoo API Script Result

Total results:12637117
1:thailand:http://media-content.flixya.com.s3.amazonaws.com/files/happyfinance721828.jpg?AWSAccessKeyId=1TKE66PETJJHG8051M02&Expires=2113671996&Signature=
2:250218458 dd2ce56969
jpg:http://uhaweb.hartford.edu/LAW/Image/250218458_dd2ce56969.jpg
3:The Big Buddha Koh Samui Samui Island Thailand
jpg:http://www.wallcoo.com/human/2009_Travel_Geographic_Desktop_02/images/
4:thailand:http://geries.files.wordpress.com/2009/03/thailand.jpg?w=500&h=348
5:thailand:http://thailandtravel1.files.wordpress.com/2009/04/thailand.jpg?w=470&h=352
6:thailand 1173192244 100357
jpg:http://dromhem.blogg.se/images/2007/thailand_1173192244_100357.jpg
7:Thailand floatingmarket
JPG:http://web.mit.edu/kjforbes/www/Pictures/Thailand-floatingmarket.JPG
8:IL Thailand 086 jpg:http://www.seal-superyachts-asia.com/gallery/thailand/photos/IL-Thailand-086.jpg

Script Result After Clicking URL



Agenda

- What is JSON?
- JSON vs. XML
- JSON Tutorial
- JSON and AJAX
- Using JSON with Yahoo! Web Services
- **Using JSON with Google Data Protocol**

Using JSON in the Google Data Protocol

- The feed is represented as a JSON object; each nested element or attribute is represented as a name/value property of the object.
- Attributes are converted to String properties.
- Child elements are converted to Object properties.
- Elements that may appear more than once are converted to Array properties.
- Text values of tags are converted to $t properties

Sample Google Data in JSON

{
"version": "1.0", "encoding": "UTF-8", "feed": { "xmlns": "http://www.w3.org/2005/Atom",
"xmlns$openSearch": "http://a9.com/-/spec/opensearchrss/1.0/", "xmlns$gd": "http:
//schemas.google.com/g/2005", "xmlns$gCal": "http://schemas.google.com/gCal/2005",
"id": {"$t": "..."}, "updated": {"$t": "2006-11-12T21:25:30.000Z"}, "title": { "type": "text", ...

Requesting and Using JSON Feeds

- Atom is Google Data's default format. If you don't specify an alt parameter in your request, then you receive an Atom feed
- To request a response in JSON format, use the alt=json parameter.
- For example, to request Google's developer calendar feed in JSON format, send the following query:
- http://www.google.com/calendar/feeds/developer-calendar@google.com/public/full?alt=json

Getting JSON Feeds in Javascript

To request a response that wraps JSON in a script tag, use the alt=json-in-script parameter and add a callback function by adding the callback=*functionName* parameter.
http://www.google.com/calendar/feeds/developer-calendar@google.com/public/full?alt=json-in-script&callback=myFunction

Sample Script to Get JSON Feeds (1/2)

```
<h3>Upcoming Google Developer Events</h3> <div id=" agenda"></div> <script> function listEvents(root) { var feed = root.feed; var entries = feed.entry || []; var html = ['<ul>']; for (var i = 0; i < entries.length; ++i) { var entry = entries[i]; var title = entry.title.$t; var start = (entry['gd$when']) ? entry['gd$when'] [0].startTime : ""; html.push('<li>', start, ' ', title, '</li>'); }
```

Sample Script to Get JSON Feeds (2/2)

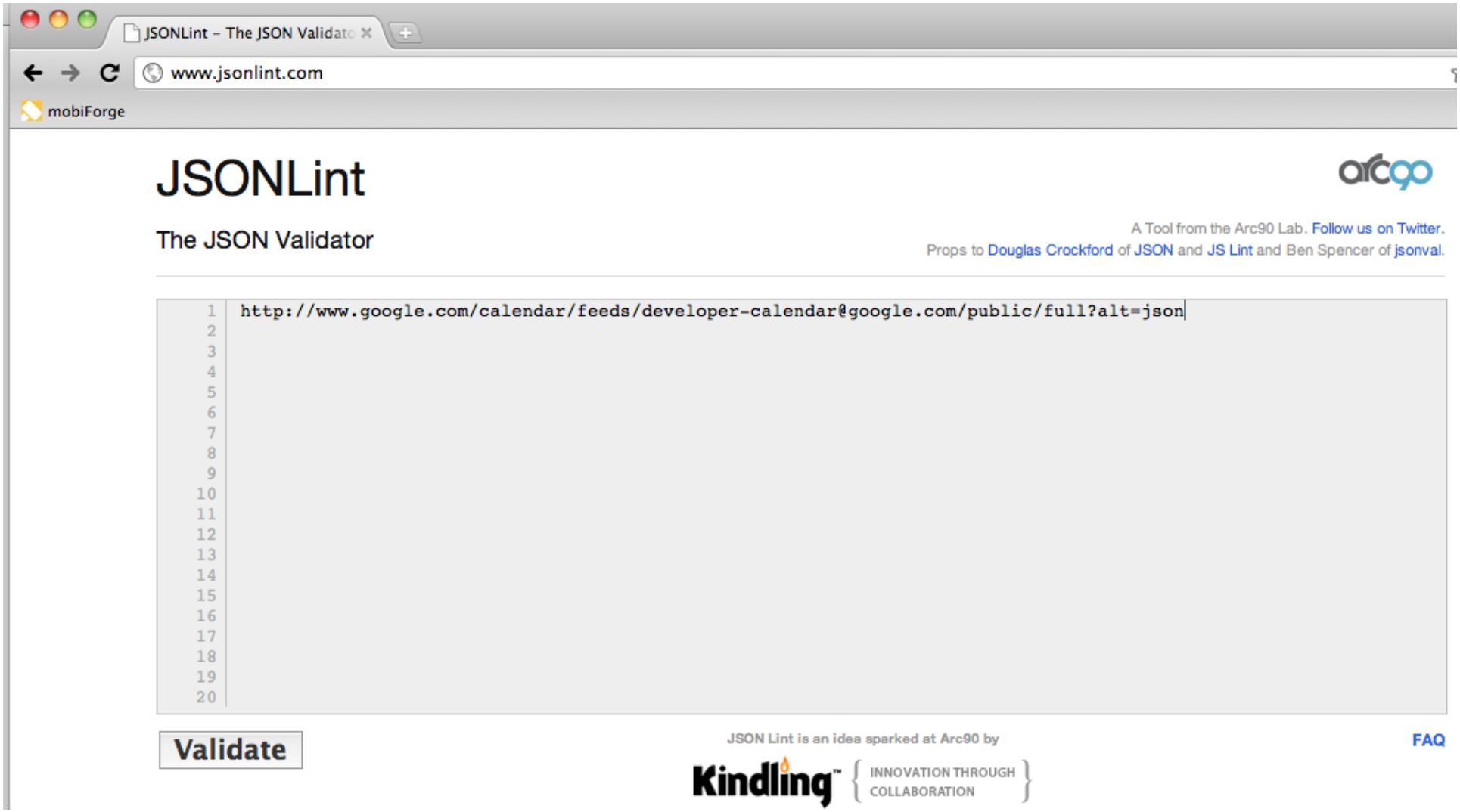
```
html.push('</ul>'); document.getElementById("agenda").
innerHTML = html.join(""); } </script> <script src="http://www.
google.com/calendar/feeds/developer-calendar@google.
com/public/full?alt=json-in-script&callback=listEvents">
</script>
```

Google Data Sample Script Result

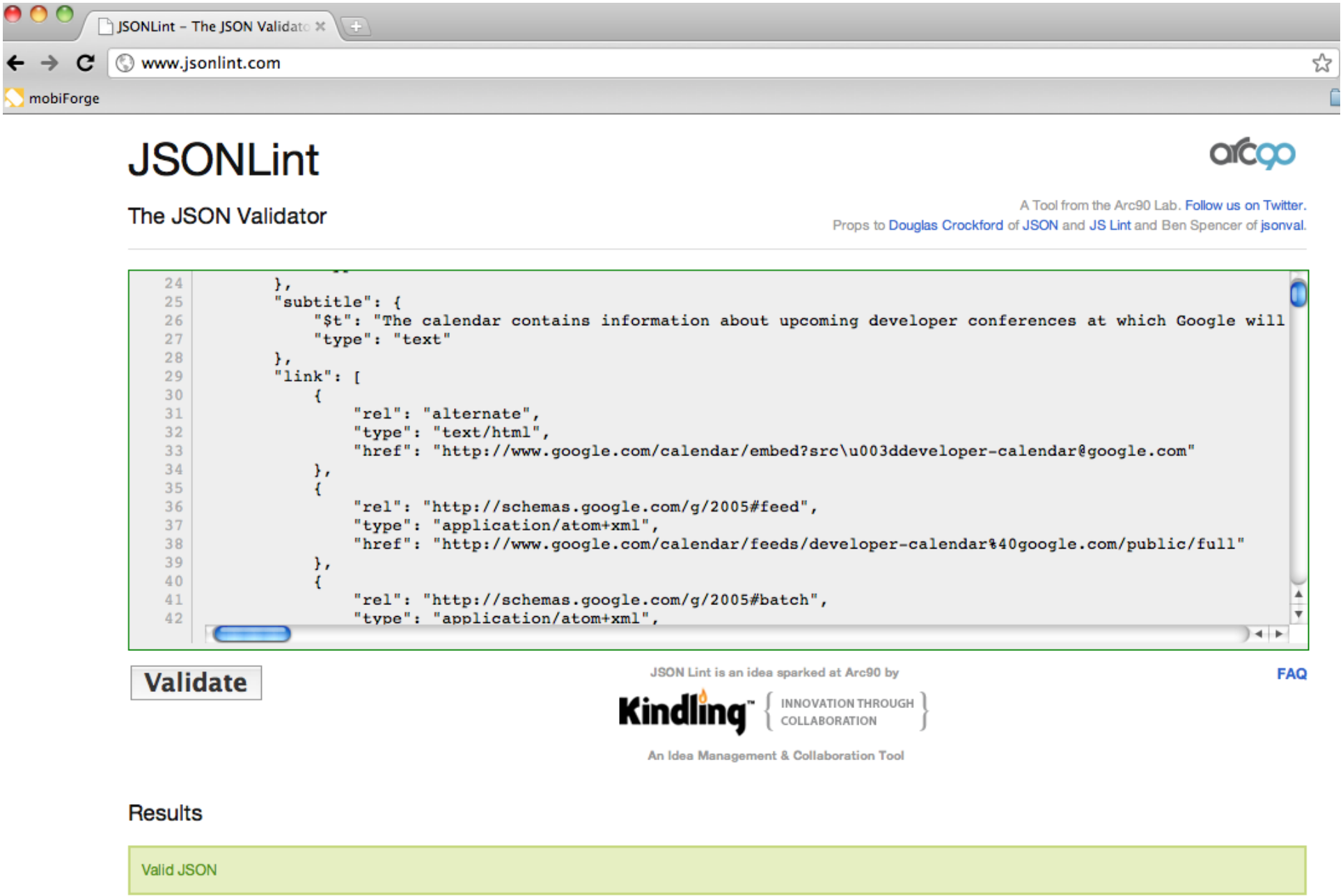
Upcoming Google Developer Events

- Google Cloud Computing for Java Developers: Platform and Monetization
- 2011-03-07 NY Journalist Hackathon
- 2011-02-10T13:00:00.000-08:00 Stanford GISSIG workshop
- 2011-02-15T01:30:00.000-08:00 Google Apps for Education UK User Group meeting
- 2011-01-19T15:00:00.000-08:00 What's New in GWT 2.1 at MCJUG
- 2011-02-21 G-Senegal
- 2011-02-18 Developer Summit 2011
- 2011-01-20T08:30:00.000-08:00 Workshop at MIT
- 2011-03-09 PyCon US
- 2011-02-02 O'Reilly Strata Conference
- 2011-01-26T15:00:00.000-08:00 GWT+GAE at SunJUG
- 2011-01-18T15:30:00.000-08:00 GWT + App Engine at NovaJUG / NCA GTUG

JSON Validator Input



JSON Validator Output Result



References

- Introducing JSON, http://www.json.org/
- JSON in JavaScript, http://www.json.org/js.html
- http://en.wikipedia.org/wiki/JSON
- http://secretgeek.net/json_3mins.asp
- http://funkatron.com/site/comments/safely-parsing-json-in-javascript/
- http://yuiblog.com/blog/2007/04/10/json-and-browser-security/
- http://www.developer.com/lang/jscript/article.php/3596836/Speeding-Up-AJAX-with-JSON.htm
- http://www.devarticles.com/c/a/JavaScript/AJAX-with-JSON/
- http://developer.yahoo.com/javascript/json.html
- http://www.jsonlint.com/