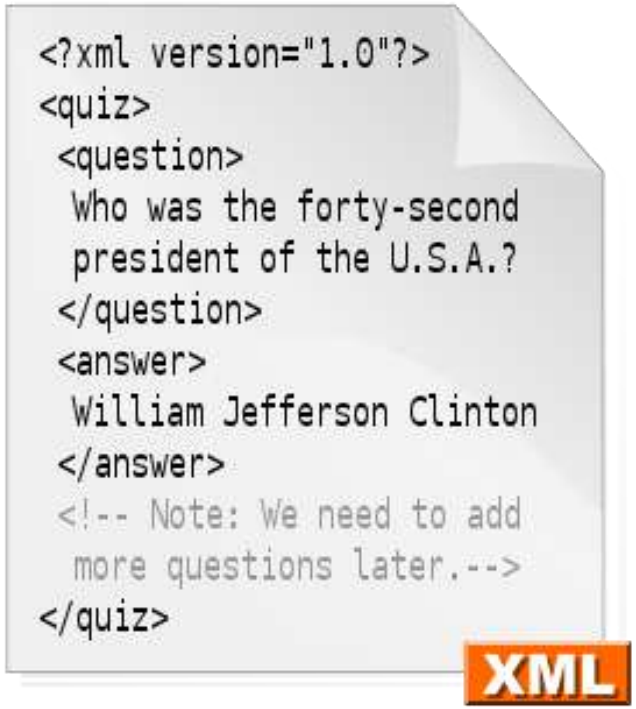
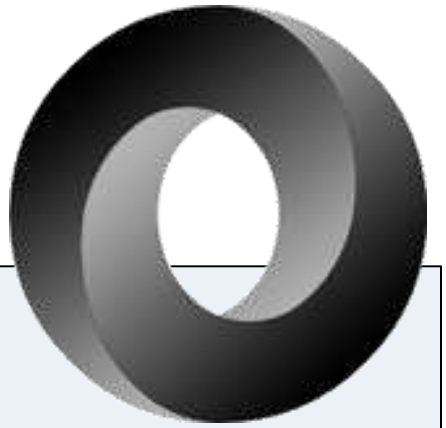
Data formats



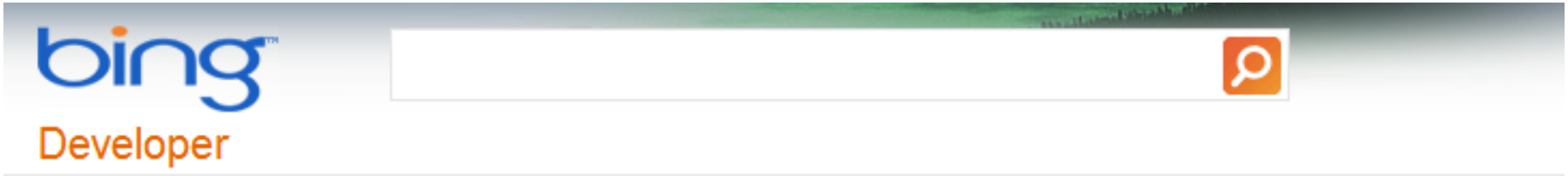


```
{
      "firstName": "John",
      "lastName" : "Smith",
      "age"      : 25,
      "address"  :
      {
          "streetAddress": "21 2nd Street",
          "city"         : "New York",
          "state"        : "NY",
          "postalCode"   : "10021"
      },
```

Overview

- Web services
  - A simple GET request (REST)
  - Or a more complicated XML message (SOAP)
- Data formats:
  - XML
  - JSON

Bing web services



Develop with Bing

Use Bing Search and Map APIs to increase your website's functionality and user appeal, or to create enterprise/consumer applications.

Sign in - Bing Search API

Sign up to use Bing Search API and create AppIDs
Learn more

Create unique search applications

- Build applications powered by Bing's technology
- Choose from multiple SourceTypes (Web, Images, Video, and More) and output protocols (JSON, SOAP, or XML)
- Customize the search results to your needs

Sign in - Bing Maps

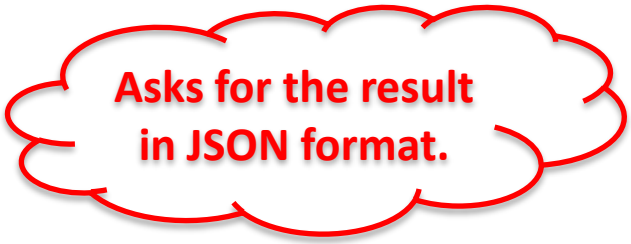
Sign up to use Bing Maps
Learn more

Create your own mapping applications

- Feature maps that render fast and deliver stunning imagery
- Easily scale from a simple static map up to the most complex interactive spatial application
- Build for mobile using our tools for Windows Phone 7, Android, and iOS

Using Bing search API

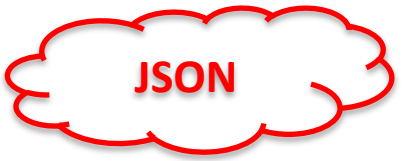
- Find the top-10 Bing results for "orediggers"
  - Make a HTTP GET request
  - `http://api.bing.net/json.aspx?`
    `AppId=AFKJEAWKFJEAWKFJA&Version=2.2&`
    `Market=enUS&`
    `Query=orediggers&Sources=web+spell&`
    `Web.Count=2&`
    `JsonType=raw`



Using Bing search API

- Find the top-10 Bing results for "orediggers"
  - Make a HTTP GET request
  - `http://api.bing.net/xml.aspx?`
    `AppId=AFKJEAWKFJEAWKFJA&Version=2.2&`
    `Market=enUS&`
    `Query=orediggers&Sources=web+spell&`
    `Web.Count=2`

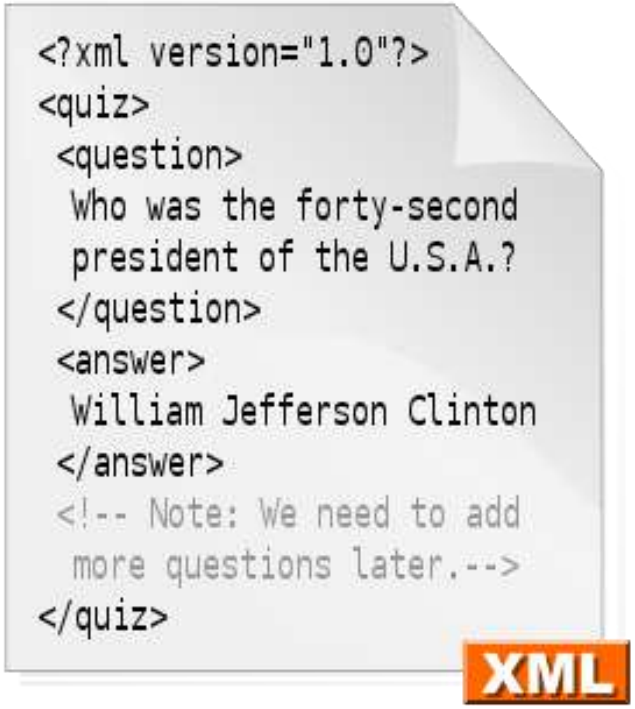

```
<?xml version="1.0" encoding="UTF-8"?>
<?pageview_candidate ?>
- <SearchResponse Version="2.2"
xmlns="http://schemas.microsoft.com/LiveSearch/2008/04/XML/element">
  - <Query>
      <SearchTerms>orediggers</SearchTerms>
    </Query>
  - <web:Web xmlns:web="http://schemas.microsoft.com/LiveSearch/2008/04/XML/web">
      <web:Total>53300</web:Total>
      <web:Offset>0</web:Offset>
    - <web:Results>
      - <web:WebResult>
          <web:Title>Colorado School of Mines Athletics</web:Title>
          <web:Description>Official site of the Orediggers with scores, statistics, pictures, and
            rosters.</web:Description>
          <web:Url>http://www.csmorediggers.com/</web:Url>
          <web:DisplayUrl>www.csmorediggers.com</web:DisplayUrl>
          <web:DateTime>2012-01-31T12:35:00Z</web:DateTime>
        </web:WebResult>
      - <web:WebResult>
          <web:Title>Montana Tech Athletics</web:Title>
          <web:Description>Official site of the Orediggers with news items, scores, statistics,
            player profiles, roster and a schedule of games.</web:Description>
          <web:Url>http://www.godiggers.com/</web:Url>
          <web:DisplayUrl>www.godiggers.com</web:DisplayUrl>
          <web:DateTime>2012-01-31T22:30:00Z</web:DateTime>
        - <web:DeepLinks>
          - <web:DeepLink>
              <web:Title>Digger Football</web:Title>
              <web:Url>http://www.godiggers.com/football/football.html</web:Url>
            </web:DeepLink>
          - <web:DeepLink>
              <web:Title>Football</web:Title>
              <web:Url>http://www.godiggers.com/football.html</web:Url>
            </web:DeepLink>
```



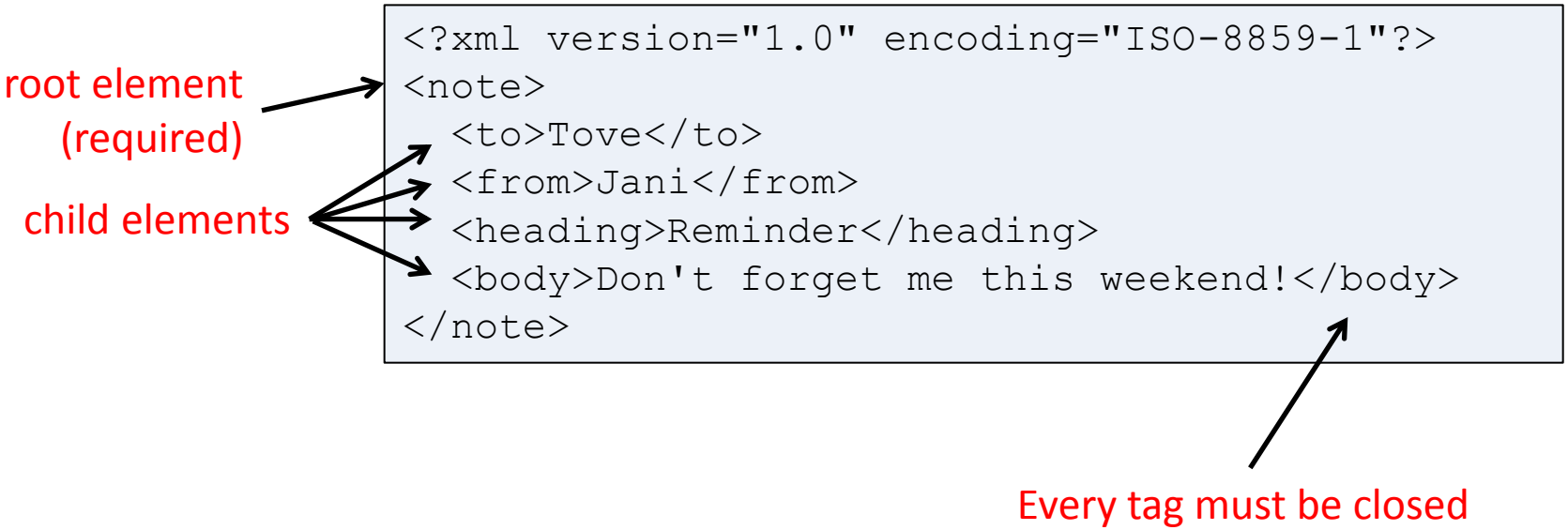

```
{
   "SearchResponse":{
      "Version":"2.2",
      "Query":{
         "SearchTerms":"orediggers"
      },
      "Web":{
         "Total":53300,
         "Offset":0,
         "Results":[
            {
               "Title":"Colorado School of Mines Athletics",
               "Description":"Official site of the Orediggers with scores, statistics, pictures, and rosters.",
               "Url":"http:\/\/www.csmorediggers.com\/",
               "DisplayUrl":"www.csmorediggers.com",
               "DateTime":"2012-01-31T12:35:00Z"
            },
            {
               "Title":"Montana Tech Athletics",
               "Description":"Official site of the Orediggers with news items, scores, statistics, player profiles,
               "Url":"http:\/\/www.godiggers.com\/",
               "DisplayUrl":"www.godiggers.com",
               "DateTime":"2012-01-31T22:30:00Z",
               "DeepLinks":[
                  {
                     "Title":"Digger Football",
                     "Url":"http:\/\/www.godiggers.com\/football\/football.html"
                  },
                  {
                     "Title":"Football",
                     "Url":"http:\/\/www.godiggers.com\/football.html"
                  },
```

Data format: XML

- XML (Extensible Markup Language)
  - Human readable
  - W3C recommendation
  - Markup language like HTML
  - Originally a document format
    - Microsoft Office (Office Open XML)
    - OpenOffice (OpenDocument)
    - Apple iWork
  - Can be used for data interchange
    - e.g. SOAP, RSS, Atom



XML




```
<b><i>This text is bold and italic</b></i>
```

Example of improperly nested HTML

```
<b><i>This text is bold and italic</i></b>
```

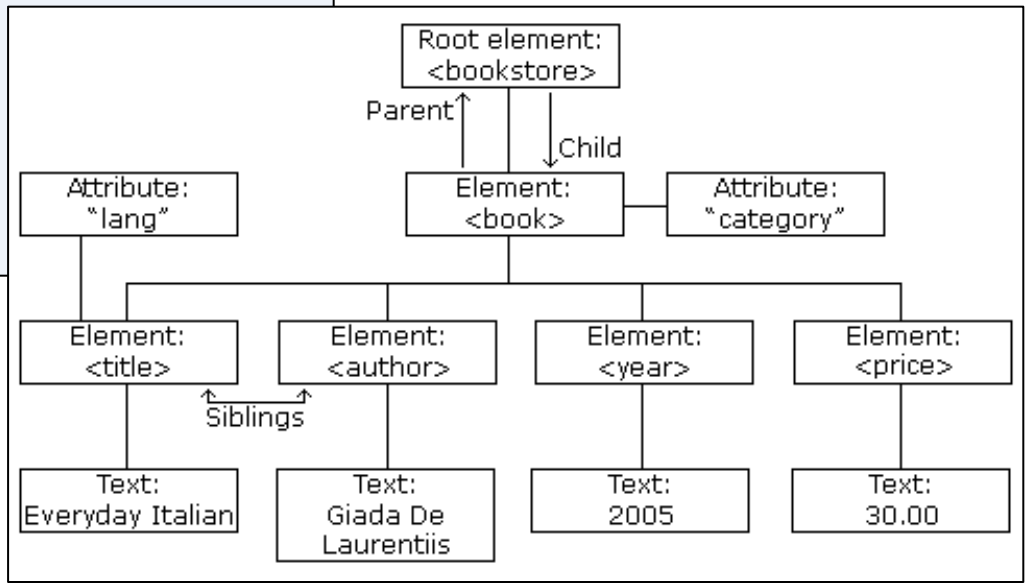
Properly nested XHTML

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
  <!-- This is a comment -->
</bookstore>
```

Tags are case-sensitive,
<title> != <Title>

Attributes must be quoted

A comment in the XML document



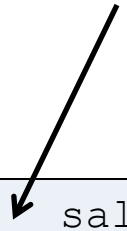
```
<p>This is a paragraph.
<br>
```

Unclosed tags in an HTML document.

```
<p>This is a paragraph.</p>
<br />
```

In XHTML, every tag must be closed.

Whitespace is preserved



```
<message>if    salary &lt; 1000 then</message>
```

Special characters need to be escaped.

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

```
<note date="10/01/2008">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

```
<note>
  <date>10/01/2008</date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

```
<note>
  <date>
    <day>10</day>
    <month>01</month>
    <year>2008</year>
  </date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Validating XML

- Method 1: Document Type Definition (DTD)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<!DOCTYPE note
[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

Validating XML

- Method 2: XML Schema
  - XSD (XML Schema Definition)

```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
    <xs:element name="to"      type="xs:string"/>
    <xs:element name="from"    type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body"    type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

Parsing XML

- DOM (Document Object Model)
  - Expressing and interacting with XML data
  - Modern browsers/languages have a built-in DOM parser
    - libxml2 - Bindings for C++, C#, Python, PHP, Perl, …
    - MSXML - Microsoft XML Core Services
  - Loads whole XML tree
- SAX (Simple API for XML)
  - Sequential event drive API
- Simple string parsing
  - Find stuff between start and end tag
  - Probably naughty… but fast…

```
<html><body><h1>W3Schools Internal Note</h1>
<div>
<b>To:</b> <span id="to"></span><br />
<b>From:</b> <span id="from"></span><br />
<b>Message:</b> <span id="message"></span>
</div>

<script type="text/javascript">
if (window.XMLHttpRequest)
{ // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp = new XMLHttpRequest();
}
else
{ // code for IE6, IE5
  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.open("GET","note.xml",false);
xmlhttp.send();
xmlDoc = xmlhttp.responseXML;

document.getElementById("to").innerHTML =
    xmlDoc.getElementsByTagName("to")[0].childNodes[0].nodeValue;
document.getElementById("from").innerHTML =
    xmlDoc.getElementsByTagName("from")[0].childNodes[0].nodeValue;
document.getElementById("message").innerHTML =
    xmlDoc.getElementsByTagName("body")[0].childNodes[0].nodeValue;
</script>
</body></html>
```

JavaScript fills in the HTML elements based on loading an XML object into the DOM and then picking out elements.

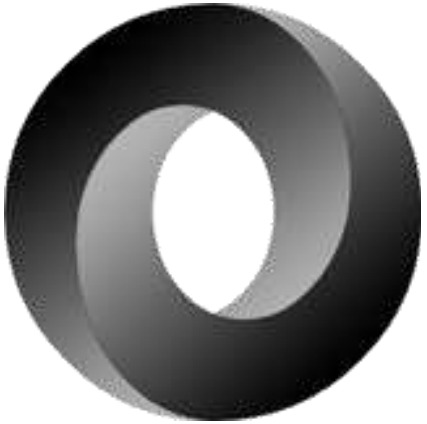
SAX parsing

```
<?xml version="1.0" encoding="UTF-8"?>
<RootElement param="value">
    <FirstElement>
        Some Text
    </FirstElement>
    <?some_pi some_attr="some_value"?>
    <SecondElement param2="something">
        Pre-Text <Inline>Inlined text</Inline> Post-text.
    </SecondElement>
</RootElement>
```

- Event sequence to SAX parser:
  - XML element start *RootElement*, attribute *param* = *value*
  - XML element start FirstElement
  - XML text node, data = "Some Text"
  - XML element end FirstElement
  - …

Data format: JSON

- JSON (JavaScript Object Notation), "Jason"
  - 2001 Popularized by Doug Crockford, Yahoo
    - http://youtu.be/-C-JoyNuQJs
  - Lightweight alternative to XML
    - Spec fits on the back of a business card
  - Human readable
  - Derived from how JavaScript represents data structures and associative arrays
  - Language independent data interchange



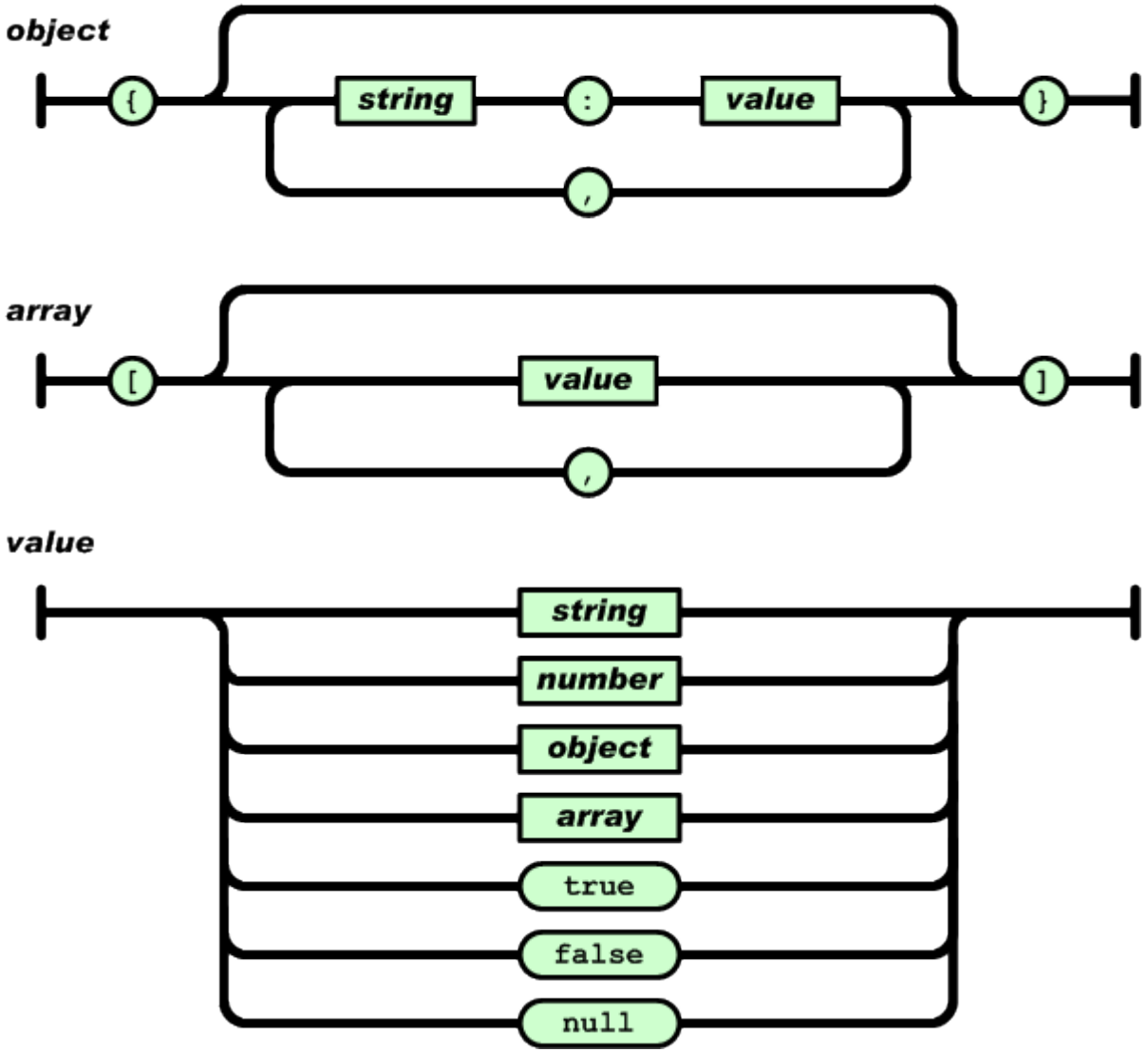
```
{to:"session", do:"test", text:"Hello world"}
```

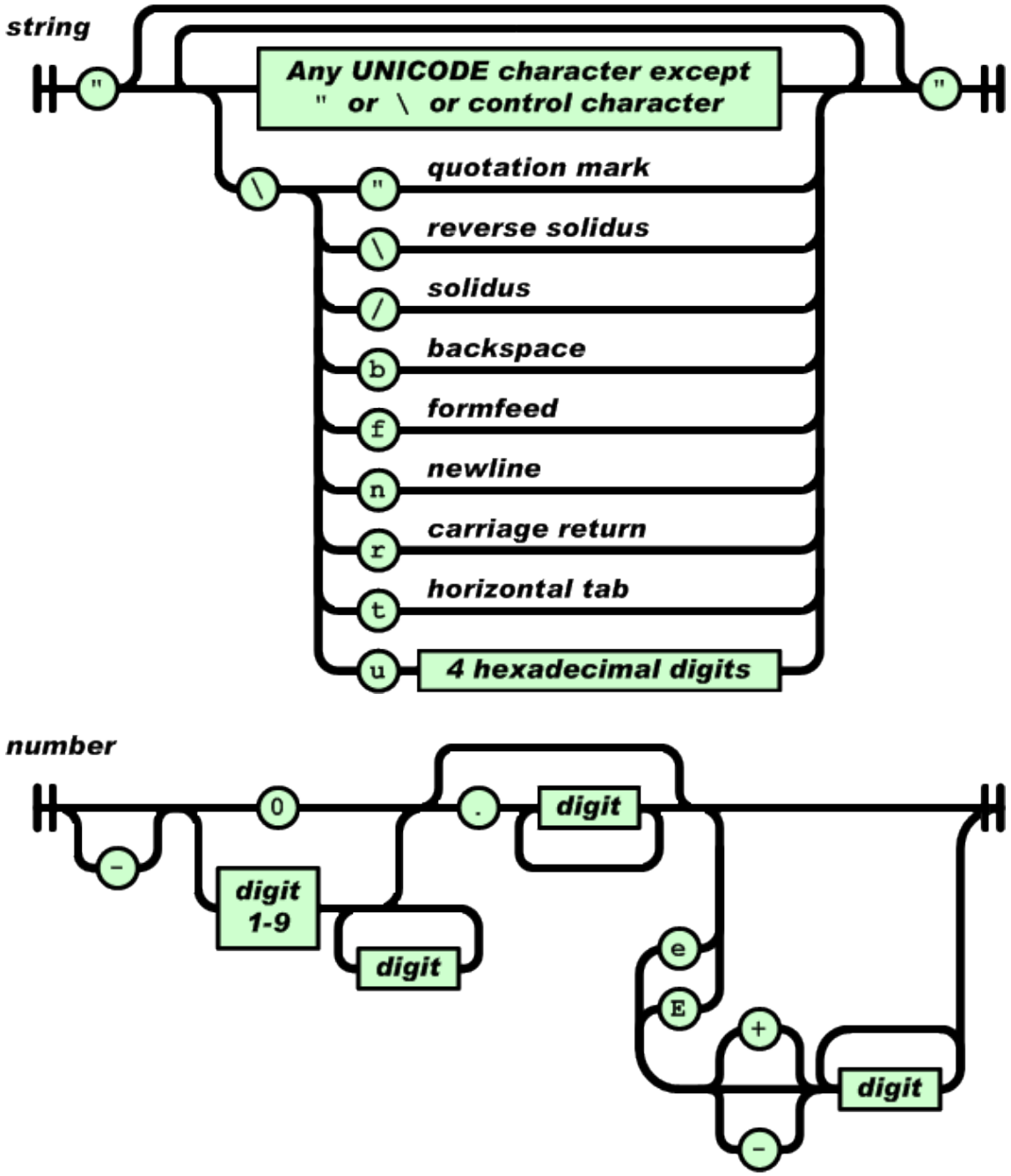
The world's first JSON message.
Failed due to JavaScript reserved word "do".
Decided to always force quoting of key.

JSON example

```
{
      "firstName": "John",
      "lastName" : "Smith",
      "age"      : 25,
      "address"  :
      {
          "streetAddress": "21 2nd Street",
          "city"         : "New York",
          "state"        : "NY",
          "postalCode"   : "10021"
      },
      "phoneNumber":
      [
          {
            "type"  : "home",
            "number": "212 555-1234"
          },
          {
            "type"  : "fax",
            "number": "646 555-4567"
          }
      ]
 }
```

JSON's grammar




string
"
Any UNICODE character except
" or \ or control character
"
\
"
quotation mark
\
reverse solidus
/
solidus
b
backspace
f
formfeed
n
newline
r
carriage return
t
horizontal tab
u
4 hexadecimal digits
number
-
0
digit
1-9
digit
.
digit
e
E
+
-
digit

Parsing JSON

- In JavaScript
  - JSON is subset of the literal JavaScript notation

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

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

Parsing JSON

- In Java
  - Freely available parsing class `JSONObject`

Method Summary	
JSONObject	accumulate(java.lang.String key, java.lang.Object value) Accumulate values under a key.
JSONObject	append(java.lang.String key, java.lang.Object value) Append values to the array under a key.
static java.lang.String	doubleToString(double d) Produce a string from a double.
java.lang.Object	get(java.lang.String key) Get the value object associated with a key.
boolean	getBoolean(java.lang.String key) Get the boolean value associated with a key.
double	getDouble(java.lang.String key) Get the double value associated with a key.
int	getInt(java.lang.String key) Get the int value associated with a key.
JSONArray	getJSONArray(java.lang.String key) Get the JSONArray value associated with a key.

Summary

- Common data formats:
  - XML
  - JSON
- Possible paper / presentation topics:
  - Analysis of differences between SOAP and REST web services
  - Learn details, do something interesting with a specific API
    - e.g. What can be done with the Facebook API?