Lect 9 – XML & JSON

Rob Capra

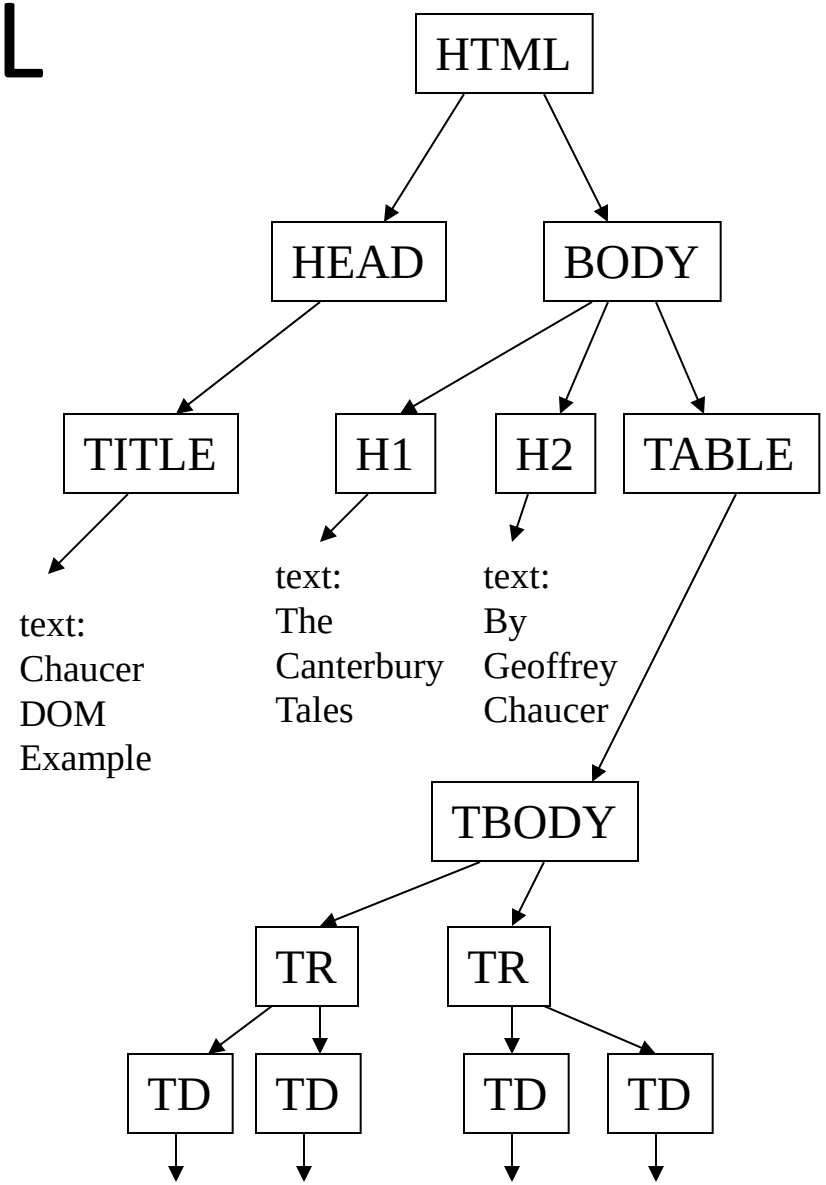
INLS 490-172

Note: Parts of these lecture notes are based on the tutorials on XML, XPath, and XSLT at W3Schools: http://www.w3schools.com/ and from the book, "XML in a Nutshell, 3rd Edition", by Harold and Means.

HTML

```
<html>
<head>
<title>Chaucer DOM Example</title>
</head>
<body>
<h1>The Canterbury Tales</h1>
<h2>by Geoffrey Chaucer</h2>
   <table>
      <tr>
        <td>Whan that Aprill</td>
        <td>with his shoures soote</td>
      </tr>
      <tr>
        <td>The droghte of March</td>
        <td>hath perced to the roote</td>
      </tr>
   </table>
<body>
</html>
```

1. Hierarchical structure
2. Common understanding of tags



EXtensible Markup Language (XML)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<release>
   <title>Punch the Clock</title>
   <artist>Elvis Costello and
      The Attractions</artist>
   <date><day>5</day>
       <month>August</month>
       <year>1983</year>
    </date>
   <label>F-Beat/Columbia</label>
</release>
```

1. Data description language
2. Hierarchical structure
3. Tags and meanings are NOT pre-defined

http://www.w3schools.com/xml/xml_whatis.asp
http://www.w3schools.com/xml/xml_syntax.asp

Valid XML Documents

- XML documents are “stricter” than HTML
  - Balanced tags/closing tags required
  - Case-sensitive
  - Proper nesting
- DTDs and XML Schemas
  - XML documents must conform to a DTD or XML Schema

XML Attributes

HTML:

```
<a href="lect10.pdf">Lect 10</a>
<form name="fred">
```

XML:

```
<fruit type="tropical">
   <name>Papaya</name>
</fruit>
```

Attributes are used less often in XML than in HTML:

```
<fruit>
   <type>tropical</type>
   <name>Papaya</name>
</fruit>
```

However, id attributes are good for having an easy way to access an element

Child Nodes and Rich Structure

```
<article>
   <title></title>
   <authors>
      <author>
         <firstname></firstname>
         <lastname></lastname>
      </author>
      <author>
         <firstname></firstname>
         <lastname></lastname>
      </author>
   </authors>
   <abstract></abstract>
   <body>
      <section>
         <subsect></subsect>
      </section>
      <section></section>
   </body>
   <references>
      <reference>
         <authors></authors>
         <title></title>
         <publication>
         </publication>
         <year></year>
      </reference>
   </references>
</article>
```

XML Examples

- SVG
  - http://www.w3schools.com/svg/svg_example.asp
  - http://www.w3schools.com/svg/svg_inhtml.asp
- RDF
  - http://en.wikipedia.org/wiki/Resource_Description_Framework
- RSS
  - http://en.wikipedia.org/wiki/RSS_%28file_format%29
- VoiceXML
  - http://www.w3.org/TR/voicexml20/#dml2.2

XML Example

```
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```

Node Relationships

1. Parent
2. Children
3. Siblings
4. Ancestors
5. Decendants

Root
Tags
Attributes
Nodes

XML Parsing in Python

- Python library: xml.etree.ElementTree
- `ElementTree` represents the whole XML document
- `Element` represents a single node in the tree

From: http://docs.python.org/2/library/xml.etree.elementtree.html

XML Parsing in Python

Read and parse, then get the root element:

```
import xml.etree.ElementTree as ET

tree = ET.parse('country_data.xml')
root = tree.getroot()

for child in root:
    print child.tag, child.attrib
```

`root` is an element

Elements have:

1. A tag (tag)
2. A dictionary of attributes (attrib)
3. A text string (text)
4. A number of child elements, stored as a Python sequence

Or, can parse from a string:

```
import xml.etree.ElementTree as ET

fp = open("country_data.xml","r")
s = fp.read()
root = ET.fromstring(s)

for child in root:
    print child.tag, child.attrib
```

XML Parsing in Python

```
import xml.etree.ElementTree as ET

tree = ET.parse('country_data.xml')
root = tree.getroot()

for child in root:
    print child.tag, child.attrib

print root[0][1].text
```

```
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```

`root` is an element

Elements have:

1. A tag (tag)
2. A dictionary of attributes (attrib)
3. A text string (text)
4. A number of child elements, stored as a Python sequence

Try:

```
print root[0].tag
print root[0][1].tag
```

XML Parsing in Python

```
import xml.etree.ElementTree as ET

tree = ET.parse('country_data.xml')
root = tree.getroot()

for child in root:
    print child.tag, child.attrib

for n in root.iter('neighbor'):
    print n.attrib
```

`root.iter` is a method that will let us iterate over all the nodes in the sub-tree below the current element (its children, their children, etc.)

```
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```

XML Parsing in Python

```
import xml.etree.ElementTree as ET

tree = ET.parse('country_data.xml')
root = tree.getroot()

for child in root:
    print child.tag, child.attrib

for country in root.findall('country'):
    rank = country.find('rank').text
    name = country.get('name')
    print name, rank
```

`.findall` finds only elements with the specified tag that are direct children of the current element

`.find` will find only the first match

`.text` accesses the element's text

`.get` accesses the element's attributes

```
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```

JavaScript Object Notation (JSON)

- JSON is a textual representation of a JavaScript data object
- Is a very common “lightweight” text-data interchange format
- Is language-independent
- Is (mostly) self-describing

But aren’t we learning Python, not JavaScript?

From: http://www.w3schools.com/json/default.asp

So why are we learning JSON?

- Because a TON of data is available...
  - http://www.whitehouse.gov/developers/policy-snapshots-json-feed
  - https://developers.google.com/blogger/docs/2.0/json/using
  - http://www.ncdc.noaa.gov/cdo-web/webservices/v1/cdows_datasets
  - https://developer.walmartlabs.com/docs/read/Search_API

JSON Example #1

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

http://en.wikipedia.org/wiki/JSON

JSON Example #1

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

JSON Data types:

- Number
- String
- Boolean
- Array
- Object (key:val)

json.loads (load string)

```
import json

fp = open("ex1.json", "r")
s = fp.read()
j = json.loads(s)

print j
print

print "first = ", j['firstName']
print "last = ", j['lastName']
print "age = ", j['age']
print "city = ", j['address']['city']
for phnum in j['phoneNumbers']:
    print phnum['type'], " = ",
        phnum['number']
```

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

JSON Example #2

```
import json

fp = open("ex2.json", "r")
s = fp.read()
j = json.loads(s)

print j
print
print "Report date = ", j['reportDate']

if j['reportType'] == 'emprec':
    for emp in j['empData']:
        print
        print "first = ", emp['firstName']
        print "last = ", emp['lastName']
        print "age = ", emp['age']
        print "city = ", emp['address']['city']
        for phnum in emp['phoneNumbers']:
            print phnum['type'], " = ",
            phnum['number']
```

```
{   "reportDate": "18-Feb-214",
    "reportType": "emprec",
    "empData":
    [
        {
            "firstName": "John",
            "lastName": "Smith",
            "age": 25,
            "address": {
                "streetAddress": "21 2nd Street",
                "city": "New York",
                "state": "NY",
                "postalCode": 10021
            },
            "phoneNumbers": [
                {
                    "type": "home",
                    "number": "212 555-1234"
                },
                {
                    "type": "fax",
                    "number": "646 555-4567"
                }
            ]
        },
        {
            "firstName": "Mary",
            "lastName": "Jones",
            "age": 28,
            "address": {
                "streetAddress": "123 41st Street",
                "city": "New York",
                "state": "NY",
                "postalCode": 10021
            },
            "phoneNumbers": [
                {
                    "type": "home",
                    "number": "212 555-9876"
                },
                {
                    "type": "cell",
                    "number": "646 555-1122"
                }
            ]
        }
    ]
}
```

JSON Example #2

```
{  "reportDate": "18-Feb-214",
   "reportType": "emprec",
   "empData":
   [
      {
         "firstName": "John",
         "lastName": "Smith",
         "age": 25,
         "address": {
            "streetAddress": "21 2nd Street",
            "city": "New York",
            "state": "NY",
            "postalCode": 10021
         },
         "phoneNumbers": [
            {
               "type": "home",
               "number": "212 555-1234"
            },
            {
               "type": "fax",
               "number": "646 555-4567"
            }
         ]
      },
      {
         "firstName": "Mary",
         "lastName": "Jones",
         "age": 28,
         "address": {
            "streetAddress": "123 41st Street",
            "city": "New York",
            "state": "NY",
            "postalCode": 10021
         },
         "phoneNumbers": [
            {
               "type": "home",
               "number": "212 555-9876"
            },
            {
               "type": "cell",
               "number": "646 555-1122"
            }
         ]
      }
   ]
}
```

Google API (deprecated)

https://developers.google.com/web-search/docs/#The_Basics

```
'https://ajax.googleapis.com/ajax/services/search/web?v=1.0&q=Paris%20Hilton'
```

```
{"responseData": {
 "results": [
  {
   "GsearchResultClass": "GwebSearch",
   "unescapedUrl": "http://en.wikipedia.org/wiki/Paris_Hilton",
   "url": "http://en.wikipedia.org/wiki/Paris_Hilton",
   "visibleUrl": "en.wikipedia.org",
   "cacheUrl": "http://www.google.com/search?q\u003dcache:TwrPfhd22hYJ:en.wikipedia.org",
   "title": "\u003cb\u003eParis Hilton\u003c/b\u003e - Wikipedia, the free encyclopedia",
   "titleNoFormatting": "Paris Hilton - Wikipedia, the free encyclopedia",
   "content": "\[1\] In 2006, she released her debut album..."
  },
  {
   "GsearchResultClass": "GwebSearch",
   "unescapedUrl": "http://www.imdb.com/name/nm0385296/",
   "url": "http://www.imdb.com/name/nm0385296/",
   "visibleUrl": "www.imdb.com",
   "cacheUrl": "http://www.google.com/search?q\u003dcache:1i34KkqnsooJ:www.imdb.com",
   "title": "\u003cb\u003eParis Hilton\u003c/b\u003e",
   "titleNoFormatting": "Paris Hilton",
   "content": "Self: Zoolander. Socialite \u003cb\u003eParis Hilton\u003c/b\u003e..."
  },
  ...
 ],
 "cursor": {
  "pages": [
   { "start": "0", "label": 1 },
   { "start": "4", "label": 2 },
   { "start": "8", "label": 3 },
   { "start": "12","label": 4 }
  ],
  "estimatedResultCount": "59600000",
  "currentPageIndex": 0,
  "moreResultsUrl": "http://www.google.com/search?oe\u003dutf8\u0026ie\u003dutf8..."
 }
}
, "responseDetails": null, "responseStatus": 200}
```

http://ajax.googleapis.com/ajax/services/search/web?v=1.0&rsz=large&q=cats

{u'responseData': {u'cursor': {u'moreResultsUrl': u'http://www.google.com/search?oe=utf8&ie=utf8&source=uds&start=0&hl=en&q=cats', u'estimatedResultCount': u'37800000', u'searchResultTime': u'0.13', u'resultCount': u'37,800,000', u'pages': [{u'start': u'0', u'label': 1}, {u'start': u'8', u'label': 2}, {u'start': u'16', u'label': 3}, {u'start': u'24', u'label': 4}, {u'start': u'32', u'label': 5}, {u'start': u'40', u'label': 6}, {u'start': u'48', u'label': 7}, {u'start': u'56', u'label': 8}], u'currentPageIndex': 0}, u'results': [{u'GsearchResultClass': u'GwebSearch', u'visibleUrl': u'en.wikipedia.org', u'titleNoFormatting': u'Cat - Wikipedia, the free encyclopedia', u'title': u'Cat - Wikipedia, the free encyclopedia', u'url': u'http://en.wikipedia.org/wiki/Cat', u'cacheUrl': u'http://www.google.com/search?q=cache:NqcQAYHQn6YJ:en.wikipedia.org', u'unescapedUrl': u'http://en.wikipedia.org/wiki/Cat', u'content': u'The domestic cat (Felis catus or Felis silvestris catus) is a small, usually furry, \ndomesticated, and carnivorous mammal. It is often called the housecat when kept\n\xa0...'},

JSON Google Search Example

```
import json
import urllib2

url =
"http://ajax.googleapis.com/ajax/services/search/web?v=1.0&rsz=large&q=cats"
fp = urllib2.urlopen(url)
html = fp.read()
myjson = json.loads(html)

print myjson

for result in myjson['responseData']['results']:
    print result
    print

count = 0
for result in myjson['responseData']['results']:
    if result['GsearchResultClass'] == 'GwebSearch':
        count += 1
        print count,
        print "title = ", result['titleNoFormatting']
        print "  url = ", result['unescapedUrl']
        print "  snippet = ", result['content']
        print
        print
```

{u'**responseData**': {u'cursor': {u'moreResultsUrl': u'http://www.google.com/search?oe=utf8&ie=utf8&source=uds&start=0&hl=en&q=cats', u'estimatedResultCount': u'37800000', u'searchResultTime': u'0.13', u'resultCount': u'37,800,000', u'pages': [{u'start': u'0', u'label': 1}, {u'start': u'8', u'label': 2}, {u'start': u'16', u'label': 3}, {u'start': u'24', u'label': 4}, {u'start': u'32', u'label': 5}, {u'start': u'40', u'label': 6}, {u'start': u'48', u'label': 7}, {u'start': u'56', u'label': 8}], u'currentPageIndex': 0}, u'**results**': [{u'**GsearchResultClass**': u'**GwebSearch**', u'visibleUrl': u'en.wikipedia.org', u'**titleNoFormatting**': u'Cat - Wikipedia, the free encyclopedia', u'title': u'Cat - Wikipedia, the free encyclopedia', u'url': u'http://en.wikipedia.org/wiki/Cat', u'cacheUrl': u'http://www.google.com/search?q=cache:NqcQAYHQn6YJ:en.wikipedia.org', u'**unescapedUrl**': u'http://en.wikipedia.org/wiki/Cat', u'**content**': u'The domestic cat (Felis catus or Felis silvestris catus) is a small, usually furry, \ndomesticated, and carnivorous mammal. It is often called the housecat when kept\n\xa0...'},