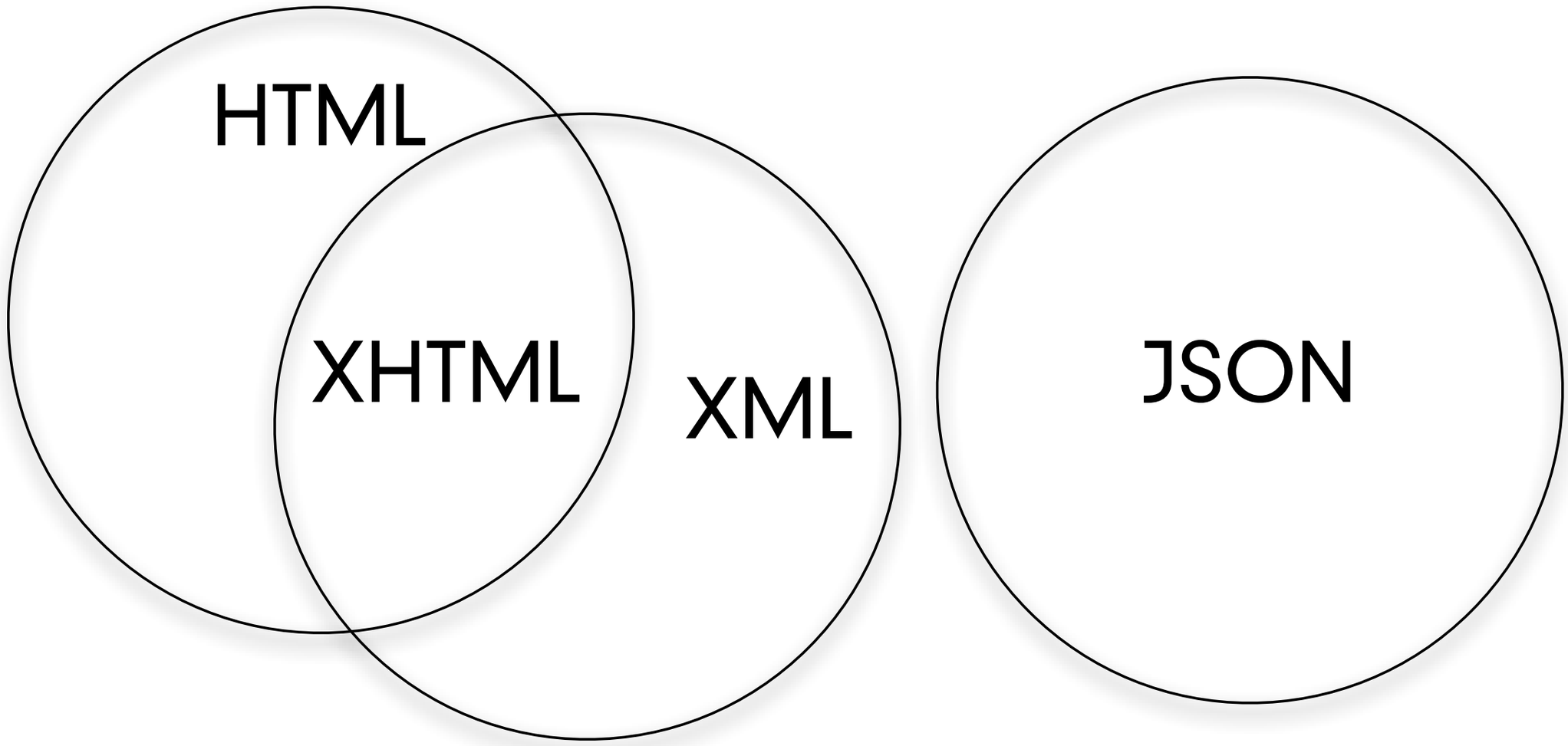
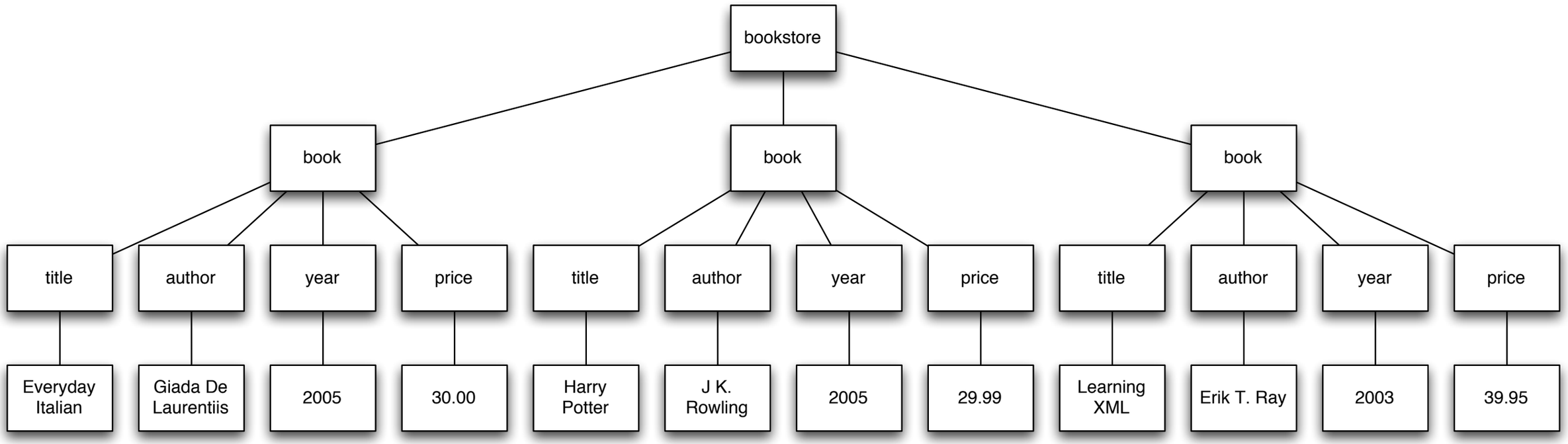
User Interaction: XML and JSON

Asst. Professor Donald J. Patterson
INF 133 Fall 2010

Web-based Data Exchange Formats



- HTML, XML and JSON
  - Structured data formats that evolved with the web
  - Text with a syntax applied
  - They can represent a huge variety of information
- They enable data transport
  - They are standardized




```
<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>
</bookstore>
```

What does it mean for XML to be well-formed?

Schema

Tags

Characters

- XML is represented as text
  - Encoding is specified in the first line (e.g.Unicode/"UTF-8")
  - Encoding describes the mapping between bits and letters

```
<?xml version="1.0" encoding="UTF-8"?>
```

- Special characters:
  - If you put a "<" in your data it will mess up XML parsing
    - <message>if salary < 1000 then</message>
  - So 5 characters are special
    - <, >, &, ' , "
    - < , > , & , ' , "
    - <message>if salary < 1000 then</message>

- Comments in XML
  - <!-- This is a comment -->
- White-space is preserved
  - <message>There is a lot of space</message>

Schema

Tags

Characters

- XML Tags are Case Sensitive
  - <Message>This is incorrect</message>
  - <message>This is correct</message>
  - <Message>This is correct</Message>

- All XML Elements Must Have a Closing Tag
- HTML
  - This is a paragraph
  - This is another paragraph
- XML and HTML and XHTML
  - This is a paragraph
  - This is another paragraph

- XML Elements Must be Properly Nested
  - HTML might have this
    - This text is bold and italic
- Valid XML requires this:
  - This text is bold and italic

- XML tags with no content may be abbreviated
  - <bookstore></bookstore>
  - <bookstore/>

- <img src="http://foo.com/picture.jpg"></img>
-

- XML tags may have attributes that describe the tag
- XML attribute values must be quoted
  - Invalid:

```
<note date=12/11/2007>
  <to>Tove</to>
  <from>Jani</from>
</note>
```

  - Valid:

```
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

Schema

Tags

Characters

- XML Documents must have a root element (This is the top-level tag)
  - <root>
    - <child>
      - <subchild>.....</subchild>
    - </child>
    - <child>
      - <subchild>.....</subchild>
    - </child>
  - </root>

- From a schema design perspective, attributes and sub-tags are pretty interchangeable

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

```
<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>
```

Well-formed XML - Schema

- Otherwise a well-formed schema is described by a separate document
  - Different types are defined, for example the “DTD”

```
<!DOCTYPE bookstore [

<!ELEMENT bookstore (book+)>
<!ELEMENT book (title,author,year,(price)+)>
<!ELEMENT title (CDATA)>
<!ELEMENT author (CDATA)>
<!ELEMENT year (CDATA)>
<!ELEMENT price (CDATA)>

<!ATTLIST book category CDATA #REQUIRED>
<!ATTLIST title lang CDATA #IMPLIED>

]>
```

- When you are ready to geek out on XML you can look into....
  - XML validation
  - Namespaces
  - XSLT
    - transforms XML to HTML for viewing

JSON

- What is JSON?
  - JSON stands for “JavaScript Object Notation”
  - JSON was designed to pass data around between browsers and servers
  - JSON has no tags, only data
  - JSON has no meta-data

JSON

- JSON
  - is also structured text
  - also has a strict syntax applied
  - can also represent a huge variety of information
  - also enables data transport ...
    - ... across systems, languages, and networks
- So what does JSON look like?

JSON


```
{
   "place":[
      {
         "suggestion":"at home",
         "meta":{
            "id":"null",
            "index":0
         },
         "size":"20.0"
      }
   ],
   "activity":[
      {
         "suggestion":"working",
         "meta":{
            "id":"null",
            "index":2
         },
         "size":"10.558333333333334"
      },
      {
         "suggestion":"sleeping",
         "meta":{
            "id":"null",
            "index":3
         },
         "size":"10.0"
      }
   ],
   "other":[
      {
         "suggestion":"(do not disturb)",
         "meta":{
            "id":"null",
            "index":1
         },
         "size":"10.0"
      }
   ],
   "error":[
      "false"
   ]
}
```

JSON

- JSON also does not DO Anything
  - It is a data format
  - A program must be written to manipulate the data
    - To search the data
    - To display the data
    - To change the data

JSON

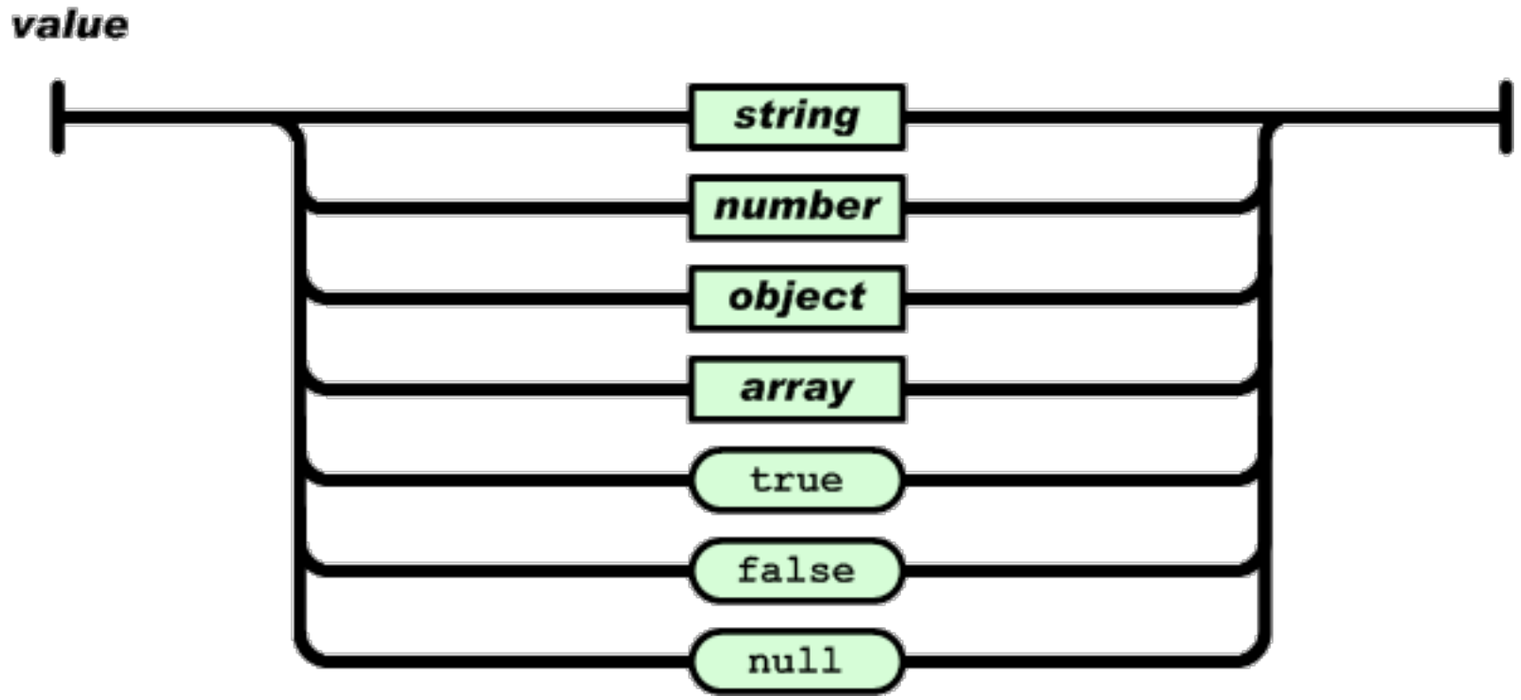
- JSON was developed by people who thought that the meta-data in XML was
  - unnecessary
  - too big
  - too hard to maintain
  - not that valuable
  - too slow
  - too much overhead to manage

What does it mean for JSON to be well-formed?

JSON

http://json.org/

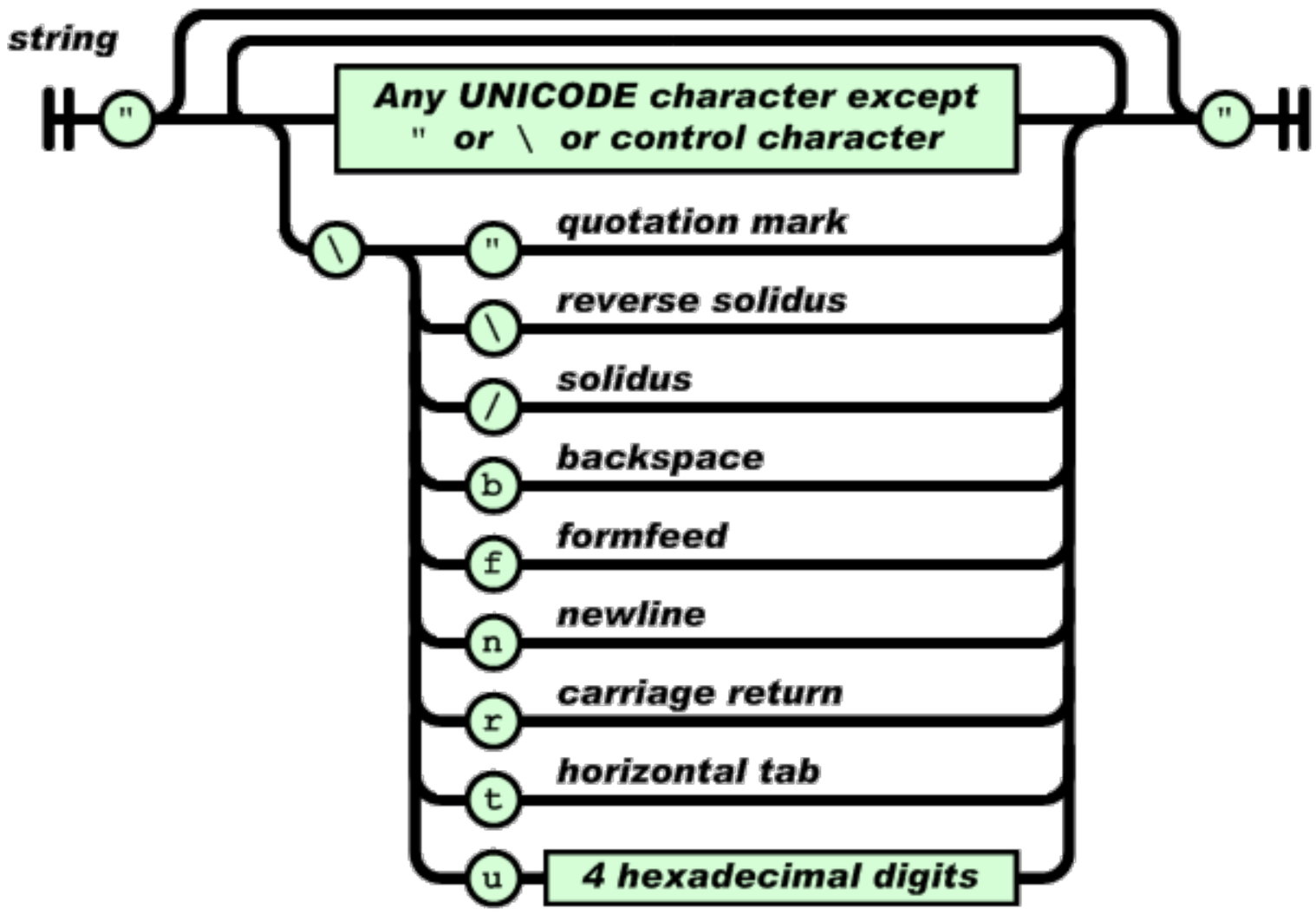
- The basic type is a value which can be
  - a string
  - a number
  - an object
  - an array
  - `true`
  - `false`
  - `null`



JSON

http://json.org/

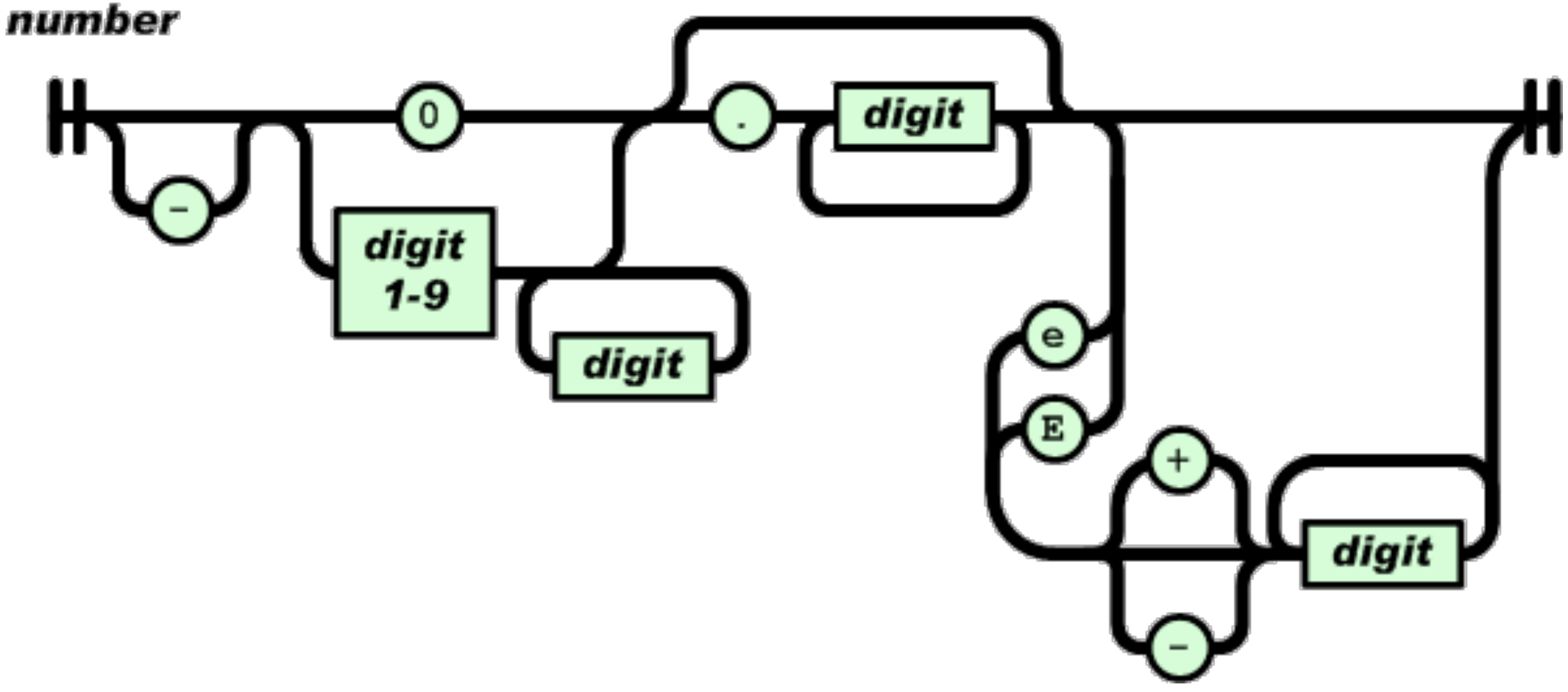
- string
  - is UNICODE
  - is always in double quotes
  - uses \ escape sequences
  - “hello world”



JSON

http://json.org/

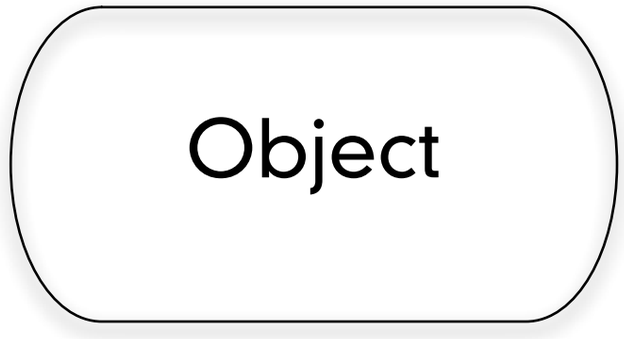
- number
  - -0.3145e1

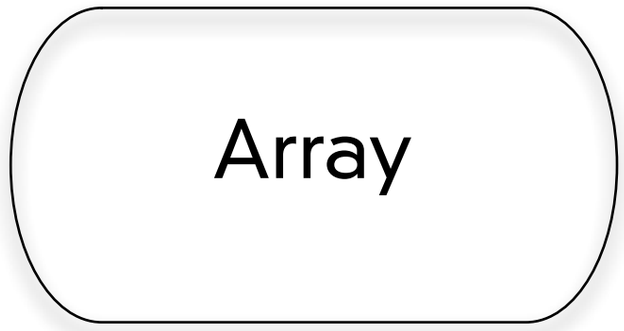


JSON

http://json.org/

- JSON has....
  - Two basic structures
    - object:
      - name/value pairs
      - like a Java “Map”
    - array
      - list of values
      - like a Java “List”

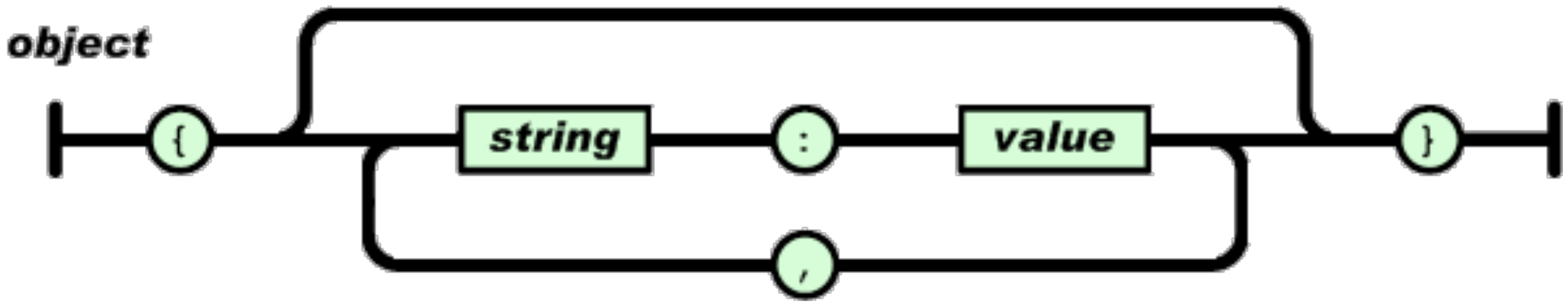




JSON

http://json.org/

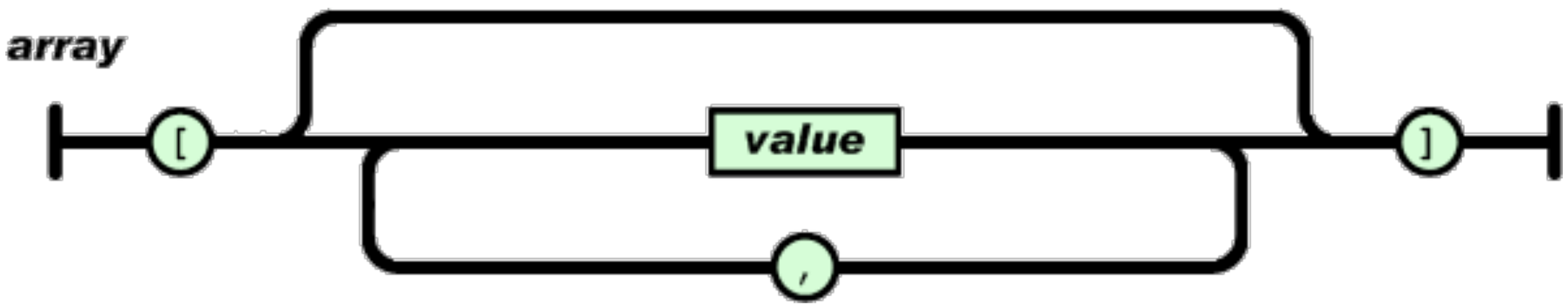
- object
  - “Map”
  - delimited by curly braces
  - name/values are separated by colons
  - elements are separated by commas
    - names are always strings
    - values are always values



JSON

http://json.org/

- array
  - “List”
  - delimited by square braces
  - elements are separated by commas
    - elements are always values



JSON

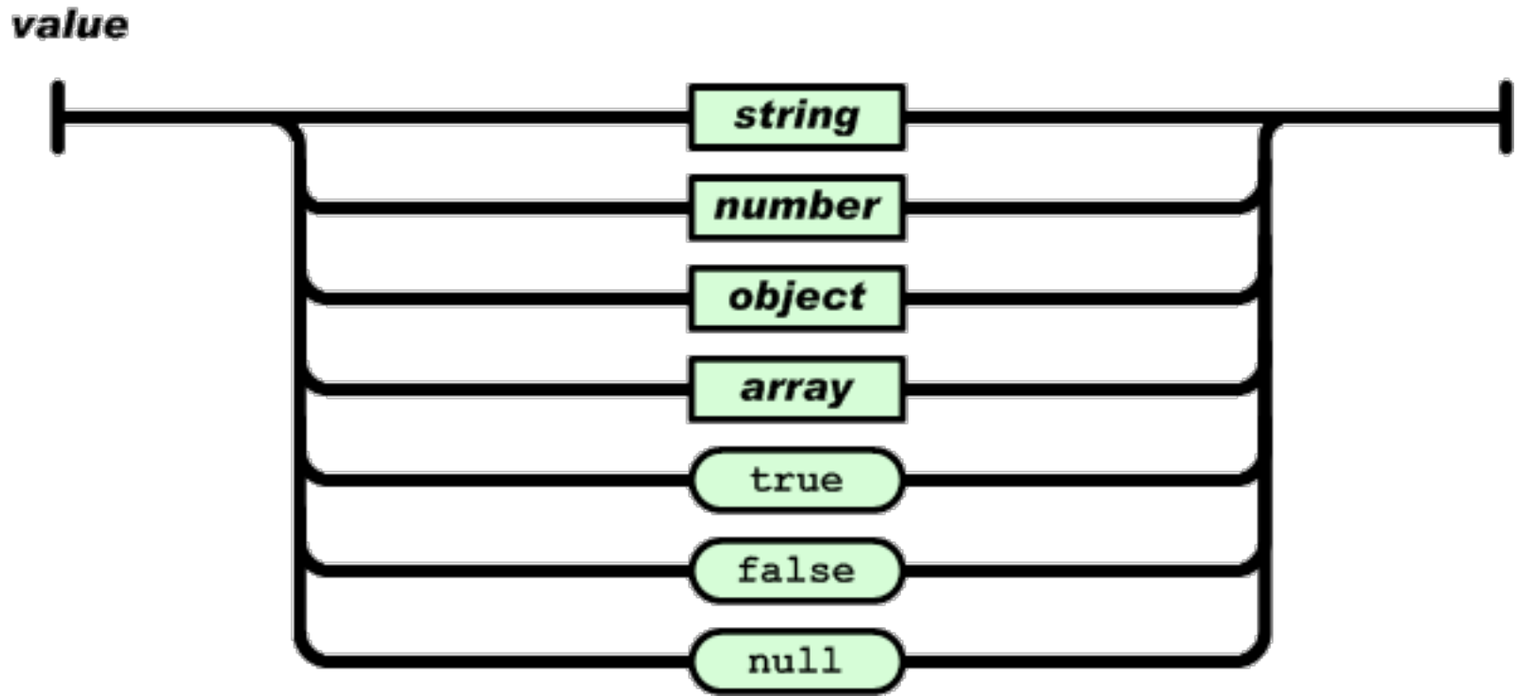
http://json.org/

- White space outside of quotes is ignored

JSON

http://json.org/

- The basic type is a value which can be
  - a string
  - a number
  - an object
  - an array
  - `true`
  - `false`
  - `null`



JSON


```
{
   "place":[
      {
         "suggestion":"at home",
         "meta":{
            "id":"null",
            "index":0
         },
         "size":"20.0"
      }
   ],
   "activity":[
      {
         "suggestion":"working",
         "meta":{
            "id":"null",
            "index":2
         },
         "size":"10.558333333333334"
      },
      {
         "suggestion":"sleeping",
         "meta":{
            "id":"null",
            "index":3
         },
         "size":"10.0"
      }
   ],
   "other":[
      {
         "suggestion":"(do not disturb)",
         "meta":{
            "id":"null",
            "index":1
         },
         "size":"10.0"
      }
   ],
   "error":[
      "false"
   ]
}
```

JSON

http://json.org/

- Supported languages
  - ASP, ActionScript, C, C++,C#,ColdFusion, D, Delphi, E, Eiffel, Erlang, Fan, Flex, Haskell, haXe, Java, JavaScript, Lasso,Lisp,LotusScript, Lua, Objective C, Objective CAML, OpenLaszlo, Perl, PHP, Pike, PL/SQL,PowerShell, Prolog, Pythong, R, Realbasic, Rebol, Ruby, Squeak, Tcl, Visual Basic, Visual FoxPro

JSON

- On beyond JSON
  - JSON validation tools are easy to find
    - http://www.jsonlint.com/
  - No defined schema language
  - No built-in namespaces (no meta-data!)
  - No built-in transformation languages

XML vs JSON

- XML is like a Ferrari
- JSON is like a good bicycle
  - A Ferrari will get you to Las Vegas faster
  - A bicycle can go off-road

- XML is beautiful and powerful
- XML is well-engineered and well-researched

- JSON is much lighter weight
- JSON is easier to just get going fast

XML vs JSON

- XML is like a Ferrari
- JSON is like a good bicycle
  - A Ferrari will get you to Las Vegas faster
  - A bicycle can go off-road

- XML is beautiful and powerful
- XML is well-engineered and well-researched

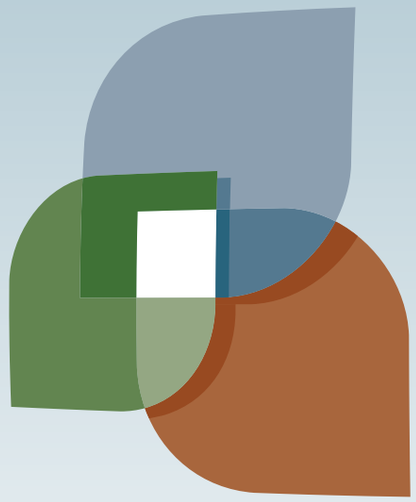
- JSON is much lighter weight
- JSON is easier to just get going fast

XML vs JSON

- XML is like a Ferrari
- JSON is like a good bicycle
  - A Ferrari will get you to Las Vegas faster
  - A bicycle can go off-road

- XML is beautiful and powerful
- XML is well-engineered and well-researched

- JSON is much lighter weight
- JSON is easier to just get going fast



LUCI

Wednesday, October 6, 2010