

- Unicode
 - Assigns a number to 110,000 letters/glyphs
 - U+0041 is an "A"
 - U+0062 is an "a"
 - U+00A9 is a copyright symbol
 - U+0F03 is an "ꠓ"
- UTF
 - Universal Character Set Transformation Format
 - describes how zeroes and ones map to Unicode characters
 - variable length
 - UTF-8 is more compact for English but can represent all characters, backward compatible with ASCII
 - UTF-16 is more compact for international use

User Interaction: XML and JSON

Assoc. Professor Donald J. Patterson
INF 133 Fall 2014



JSON

- JSON
 - also structured text
 - also with a syntax applied
 - it can also represent a huge variety of information
 - It 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"
  ]
}
```

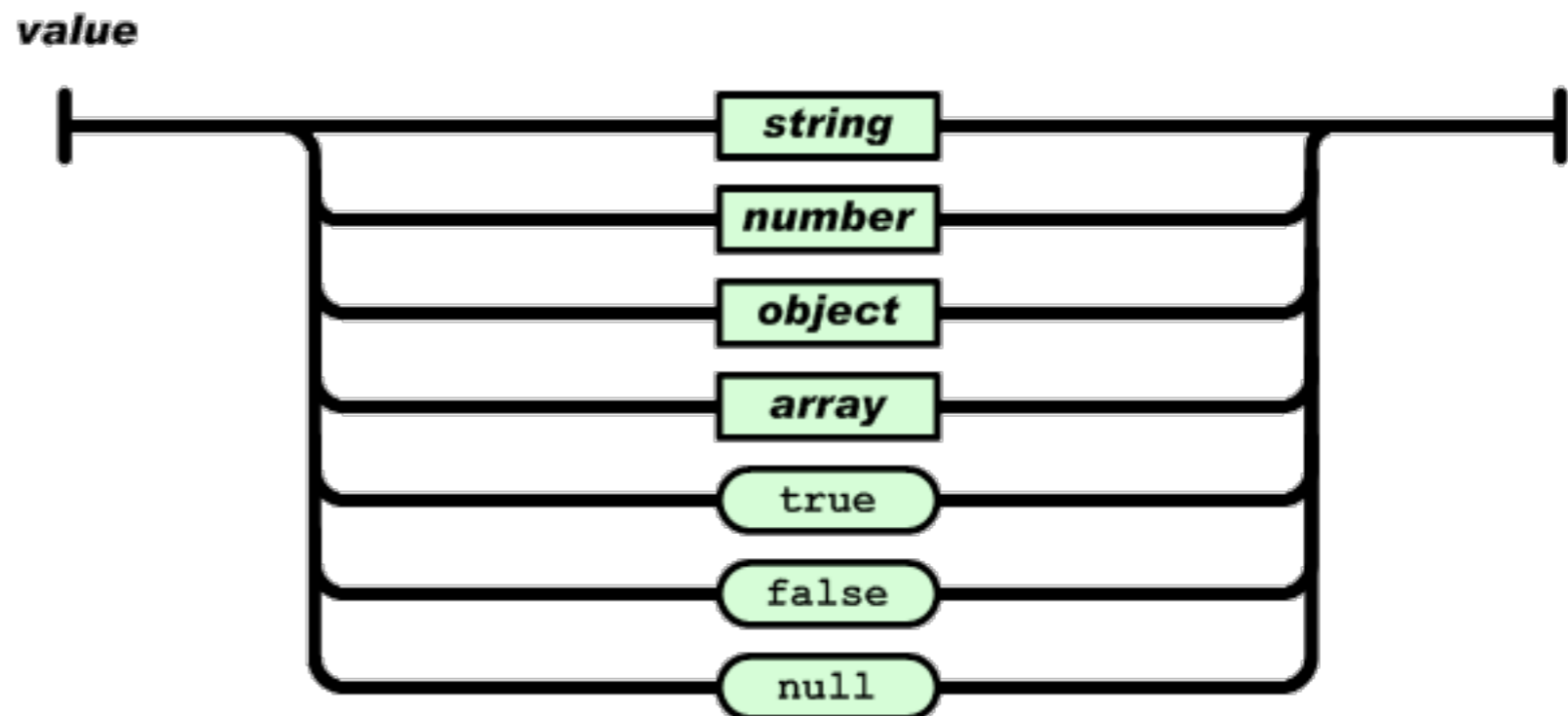
- What is JSON?
 - JSON stands for “JavaScript Object Notation”
 - JSON was designed to pass data around between browsers and servers
 - JSON has no tags, only data
 - JSON has no meta-data

- JSON 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

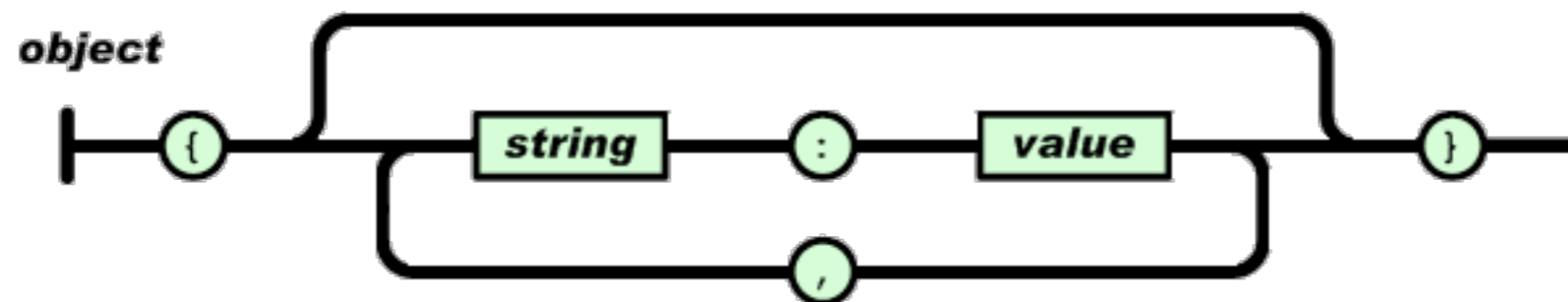
- JSON was developed by people who thought that the meta-data in XML was
 - unnecessary
 - too big
 - too hard to maintain
 - not that valuable
- It also happens to be the native data storage format in Javascript / browsers

- Details
 - Two basic structures
 - object:
 - name/value pairs
 - think Map, Associative Array or Dictionary
 - array
 - list of values
 - think List

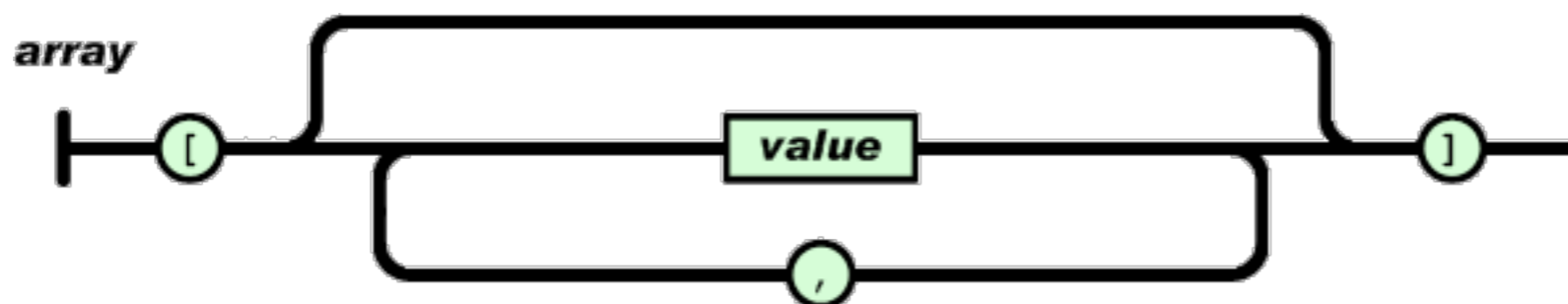
- Details
 - The basic type is a value which can be
 - a string
 - a number
 - an object
 - an array
 - **true**
 - **false**
 - **null**



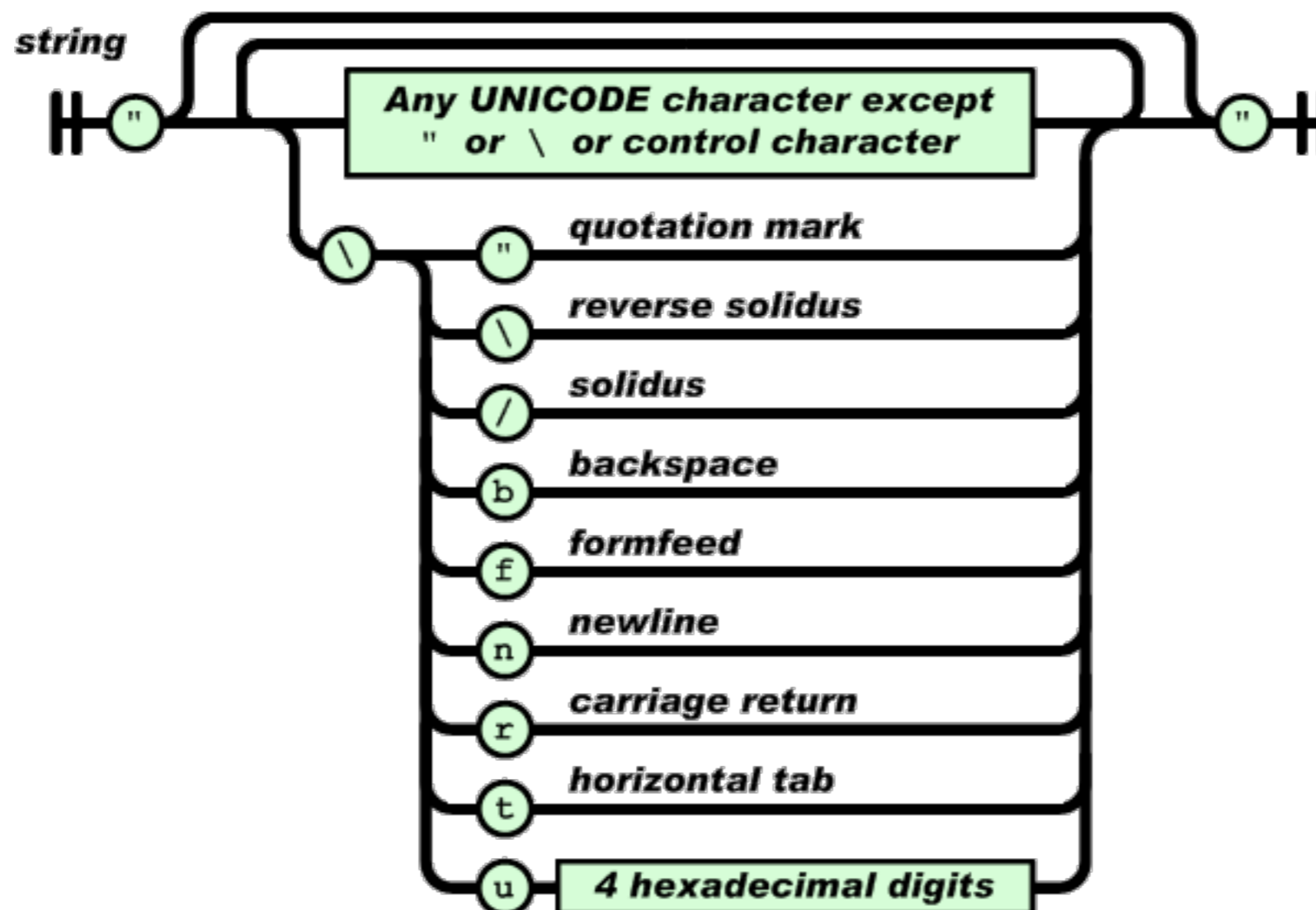
- Details
 - Object
 - Delimited by curly braces
 - name/values are separated by colons
 - elements are separated by commas
 - names are always strings
 - values are always values



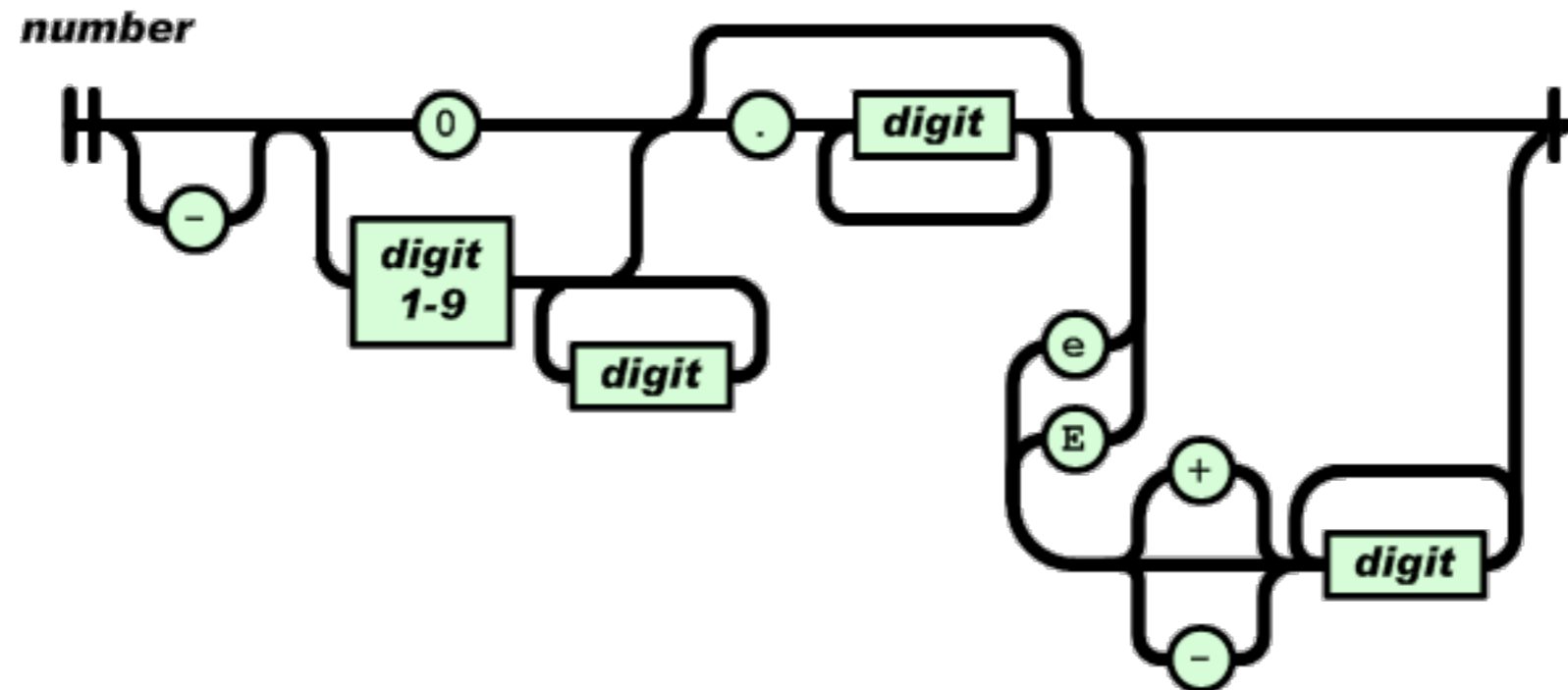
- Details
 - Array
 - Delimited by square braces
 - elements are separated by commas
 - elements are always values



- Details
 - String
 - is UNICODE, recommended is "utf-8"
 - is always in double quotes
 - uses \ escape sequences



- Details
 - Number



- Details
 - White space outside of quotes is ignored

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"
  ]
}
```


- 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, Python, R, Realbasic, Rebol, Ruby, Squeak, Tcl, Visual Basic, Visual FoxPro

- On beyond JSON
 - JSON validation tools are easy to find
 - For example, 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
 - A Ferrari will get you to Las Vegas faster
- JSON is like a good bicycle
 - 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

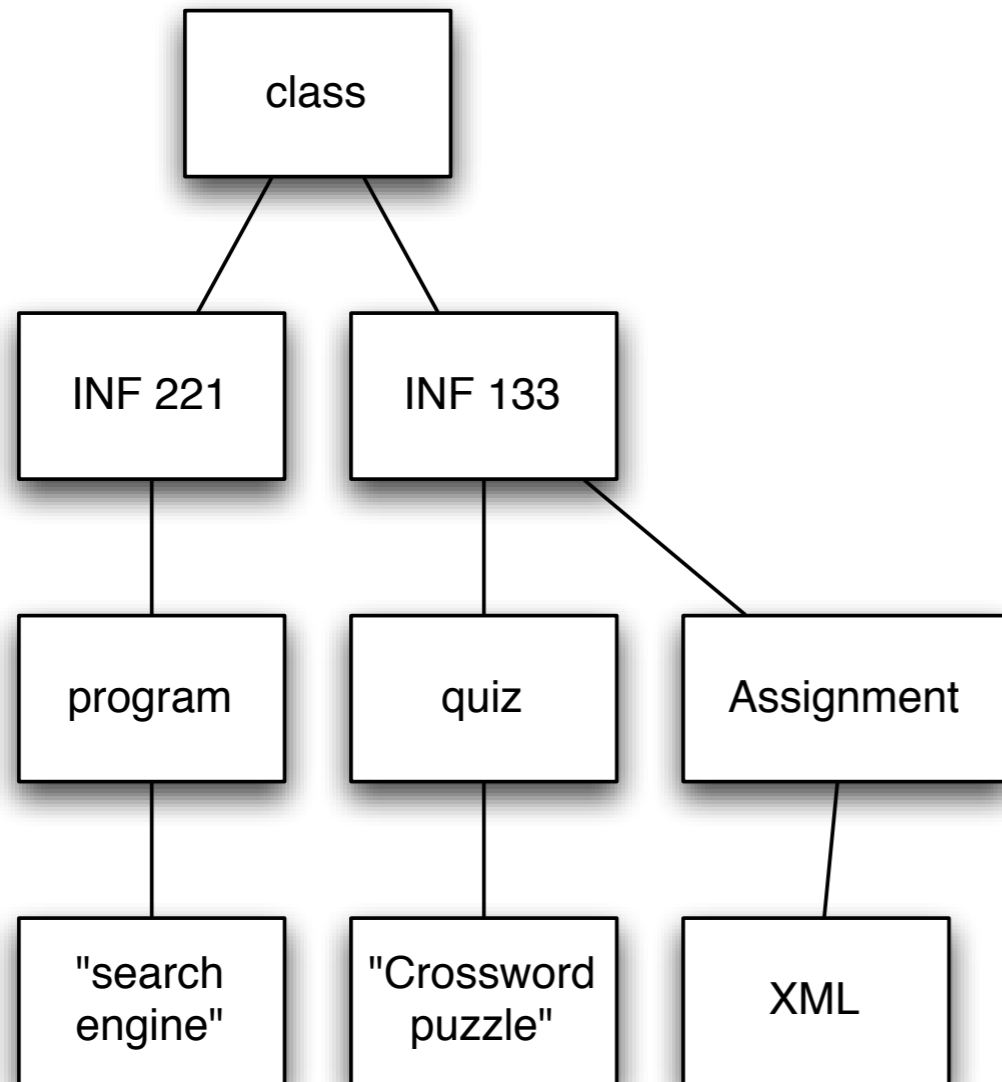


- JSON is like XML
 - They are both human-readable text
 - They are both hierarchical/ tree-structured
 - Both can be parsed and used in many languages
 - Both can be passed in AJAX requests
 - (despite the X in AJAX)

- JSON is different than XML
 - JSON does not have tags
 - JSON is less verbose
 - quicker to write
 - quicker to read
 - quicker to transport
 - JSON can be parsed trivially using the `eval()` procedure in Javascript
 - JSON has arrays, XML does not
 - XML is extensible JSON usually isn't

- Using either looks like:
 - get the JSON/XML string
 - convert it to a data structure
 - JSON -> eval(<string>)
 - XML -> some parse function (lib dependent)
 - Use the data structure
- Do not process either type of data by “hand”.
 - input: Use a library to parse the data
 - output:
 - Create the data in native data structures
 - Use a program or method to output the data structure in JSON/XML

Example



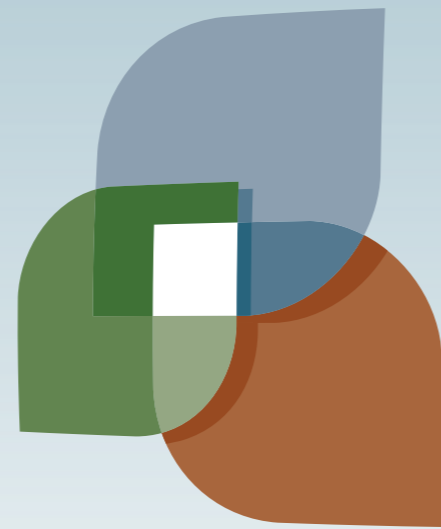
- Represent this as
 - XML
 - JSON
- There is not an absolutely correct answer to how to interpret this tree in the respective languages.
- There are multiple ways to interpret what this tree means.

Example

```
<?xml version="1.0"?>
<class>
  <INF_221>
    <program>
      search engine
    </program>
  </INF_221>
  <INF_133>
    <quiz>
      crossword puzzle
    </quiz>
    <Assignment>
      <XML/>
    </Assignment>
  </INF_133>
</class>
```

```
{
  "class": {
    "INF 221": {
      "program": "search engine"
    },
    "INF 133": {
      "quiz": "Crossword puzzle",
      "Assignment": "XML"
    }
  }
}
```


Poll Everywhere



L U C I

