XML & JSON

Background

- XML and JSON are to standard, textual data formats for representing arbitrary data
  - XML stands for “eXtensible Markup Language”
  - JSON stands for “JavaScript Object Notation”
- Both are commonly used in practice
- XML came first
- JSON, which uses JavaScript syntax, became popular for representing data in web applications and services
  - If you’re using JavaScript, JSON is an obvious choice
- Both formats are reasonable choices, although some people have strong biases
- Most programming languages have libraries for parsing and generating both XML and JSON
- You should be familiar with both

Structure of XML Documents

- Header
- Root Element
- Start Tags / End Tags
- Element Contents
  - Child Elements
  - Text
  - Both (mixed contents)
- Element Attributes
- Comments
- Entity References

- Examples
  - Element-heavy
  - Attribute-heavy
  - Hybrid

Structure of JSON Documents

- Supported data types: Objects, Arrays, Numbers, Strings, Boolean, Null
- Objects delimited by { … } with comma-separated properties in between
  - { “name”: “Bob”, “age”: 32, “alive”: true }
- Arrays delimited by […] with comma-separated elements in between
  - [“testing”, 1, 2, 3, { “gpa”: 3.4 }]
- Example

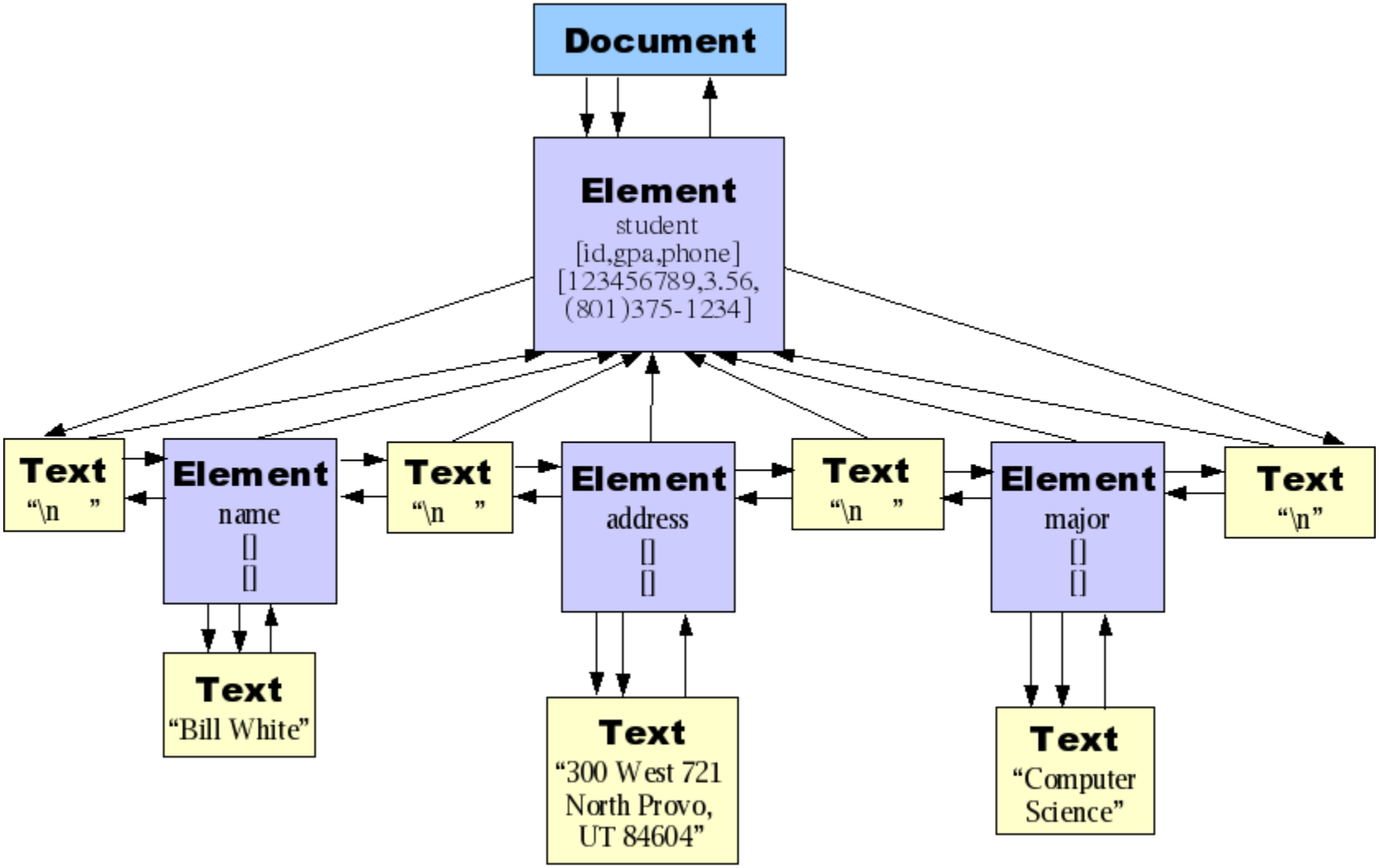
Parsing XML & JSON Data

- Most languages provide both XML and JSON parsers, so there's no need to write your own
- Two Major Types of Parsers
  - DOM Parsers
    - Convert XML or JSON text to an in-memory tree data structure (the tree is called a DOM, or "document object model")
    - After running the parser to create a DOM, traverse the DOM to extract the data you want
  - Stream Parsers
    - Tokenizers that return one token at a time from the XML or JSON data file
  - DOM parsers are easier to use, but are slower and use more memory. Pick the right parser for the job.

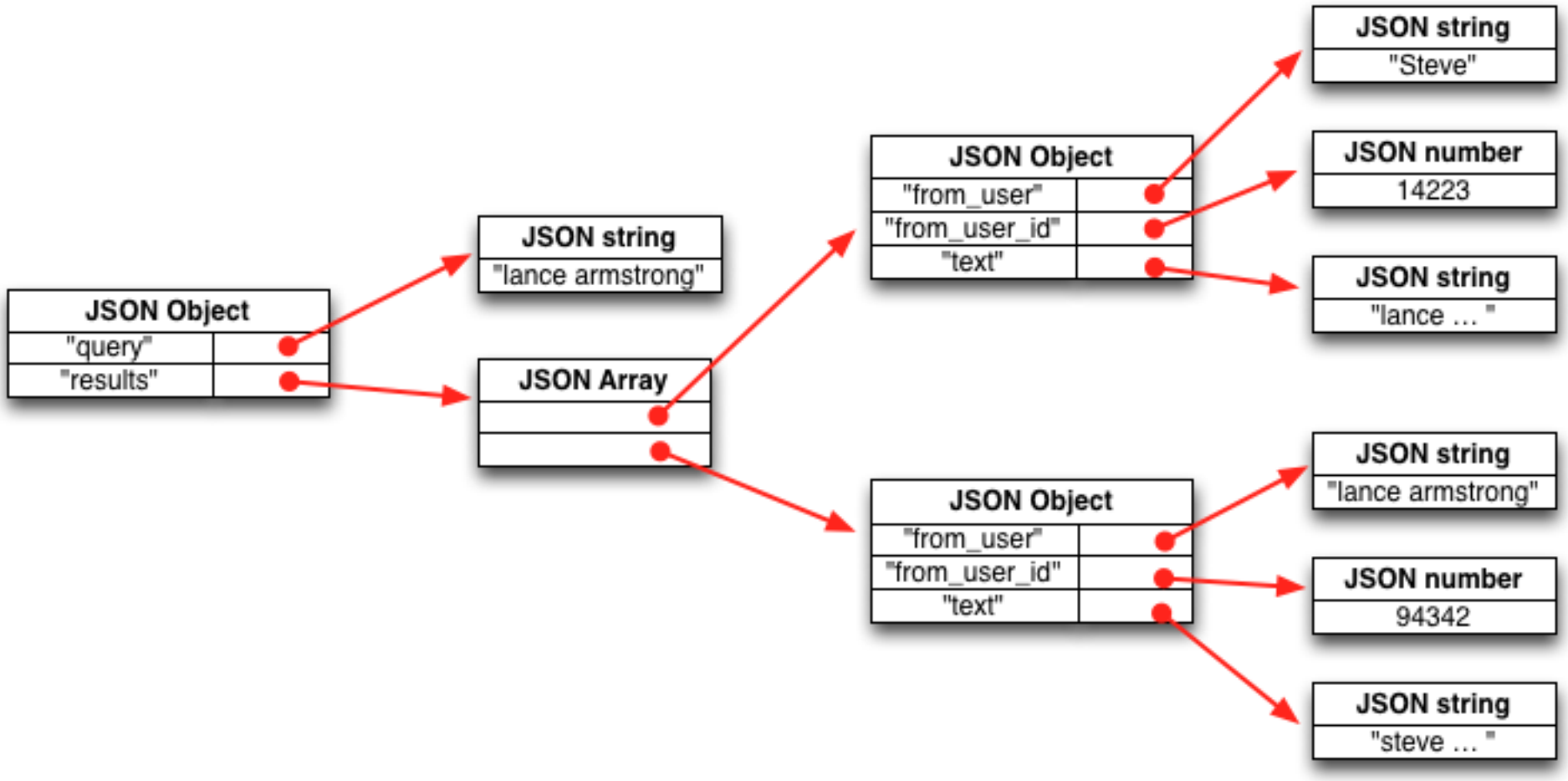
Android Parsing Examples

- DOM Parsers
  - Parse XML with DOM parser
  - Parse JSON with DOM parser
- Stream Parsers
  - Parse XML with stream parser
  - Parse JSON with stream parser

XML DOM


Document
Element
student
[id,gpa,phone]
[123456789,3.56,
(801)375-1234]
Text
"\n "
Element
name
[]
[]
Text
"\n "
Element
address
[]
[]
Text
"\n "
Element
major
[]
[]
Text
"\n"
Text
"Bill White"
Text
"300 West 721
North Provo,
UT 84604"
Text
"Computer
Science"

JSON DOM


JSON Object
"query"
"results"
JSON string
"lance armstrong"
JSON Array
JSON Object
"from_user"
"from_user_id"
"text"
JSON string
"Steve"
JSON number
14223
JSON string
"lance … "
JSON Object
"from_user"
"from_user_id"
"text"
JSON string
"lance armstrong"
JSON number
94342
JSON string
"steve … "

Generating XML & JSON Data

- Programs often need to generate (or create) XML and JSON data
- You can print XML or JSON data yourself (it's just text), but it's better to use a library (they handle tricky special cases like escaping special characters)
- Two Major Ways to Generate XML or JSON
  - Create DOM tree in memory, and then tell the tree to write itself to text
  - Use an XML or JSON "serializer" class, which lets you write one token at a time

Android Data Generation Examples

- Create and save DOM tree
  - Create XML with DOM tree
  - Create JSON with DOM tree
- Use “serializer” class
  - Create XML with serializer
  - Create JSON with serializer