Mobile Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Android & File Formats

Agenda

- Background
- Common Formats
- JSON Discussion
- Simple Example

Question?

- Given a particular set of data, how do you store it permanently?
  - What do you store on disk?
  - What format?
  - Can you easily transmit over the web?
  - Will it be readable by other languages?
  - Can humans read the data?
- Examples:
  - A Square
  - A Dictionary
  - A Donation…

Storage Using Plain Text?

- Advantages
  - Human readable (good for debugging / manual editing)
  - Portable to different platforms
  - Easy to transmit using web

- Disadvantages
  - Takes more memory than necessary

- Alternatives? - use a standardized system / Common Format
  - Makes the information more portable

Common Formats

- Comma Separated Values (CSV)
- Name/Value Pairs
- YAML
- XML
- JSON

CSV

A **comma-separated values** (**CSV**) (also sometimes called *character-separated values,* because the separator character does not have to be a comma) file stores tabular data (numbers and text) in plain-text form. Plain text means that the file is a sequence of characters, with no data that has to be interpreted instead, as binary numbers. A CSV file consists of any number of records, separated by line breaks of some kind; each record consists of fields, separated by some other character or string, most commonly a literal comma or tab. Usually, all records have an identical sequence of fields.

http://en.wikipedia.org/wiki/Comma-separated_values

```
“mocha”, “costa”, 2.0,  3.5,  0
“americano”, “costa”, 3.0, 4.5, 1
“cappucino”, “starbucks”, 4.0, 1.5, 0
```

coffees.csv

Name/Value Pairs

A **name–value pair**, **key–value pair**, **field–value pair** or **attribute–value pair** is a fundamental data representation in computing systems and applications. Designers often desire an open-ended data structure that allows for future extension without modifying existing code or data. In such situations, all or part of the data model may be expressed as a collection of tuples <*attribute name*, *value*>; each element is an attribute–value pair. Depending on the particular application and the implementation chosen by programmers, attribute names may or may not be unique.

http://en.wikipedia.org/wiki/Attribute-value_pair

```
db.url=jdbc:cloudbees://pacemaker
db.driver=com.mysql.jdbc.Driver
db.user=pacemaker
db.pass=pacemaker
jpa.ddl=create
```

application .conf

```
name="mocha"
shop="costa"
rating=3.5
price=2.0
favourite=0
id=1
```

coffees.conf

YAML

YAML (rhymes with *camel*) is a human-readable data serialization format that takes concepts from programming languages such as C, Perl, and Python, and ideas from XML and the data format of electronic mail (RFC 2822). YAML was first proposed by Clark Evans in 2001,[1] who designed it together with Ingy döt Net[2] and Oren Ben-Kiki.[2] It is available for several programming languages.

YAML is a recursive acronym for "YAML Ain't Markup Language". Early in its development, *YAML* was said to mean "Yet Another Markup Language",[3] but it was then reinterpreted (backronyming the original acronym) to distinguish its purpose as data-oriented, rather than document markup.

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

```
Coffee(c1):
    name       : mocha
    shop       : costa
    price      : 2.0
    rating     : 3.5
    favourite  : 0

Coffee(c2):
    name       : americano
    shop       : costa
    price      : 3.0
    rating     : 4.5
    favourite  : 1


Coffee(c3):
    name       : cappucino
    shop       : starbucks
    price      : 4.0
    rating     : 1.5
    favourite  : 0
```

data.yaml

XML

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

Extensible Markup Language (**XML**) is a markup language that defines a set of rules for encoding documents in aformat that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification[3] produced by theW3C, and several other related specifications,[4] all free open standards.[5]

The design goals of XML emphasize simplicity, generality, and usability over the Internet.[6] It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services.

Many application programming interfaces (APIs) have been developed to aid software developers with processing XML data, and several schema systems exist to aid in the definition of XML-based languages.

data.xml

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

<coffee objname="c1">
    <name> mocha </name>
    <shop> costa </shop>
    <price> 2.0 </price>
    <rating> 3.5</rating>
    <favourite> 0 </favourite>
</coffee>

<coffee objname="c1">
    <name> americano </name>
    <shop> costa </shop>
    <price> 3.0 </price>
    <rating> 4.5 </rating>
    <favourite> 1 </favourite>
</coffee>

<coffee objname="c1">
    <name> cappucino </name>
    <shop> starbucks </shop>
    <price> 4.0 </price>
    <rating> 1.5 </rating>
    <favourite> 0 </favourite>
</coffee>
```

JSON

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

JSON (*jay-sawn*, *jay-sun*), or **JavaScript Object Notation**, is a text-based open standard designed for human-readable data interchange. Derived from the JavaScript scripting language, JSON is a language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, JSON is language-independent, with parsers available for many languages.

The JSON format was originally specified by Douglas Crockford, and is described in RFC 4627. The official Internet media type for JSON is `application/json`. The JSON filename extension is `.json`.

The JSON format is often used for serializing and transmitting structured data over a network connection. It is used primarily to transmit data between a server and web application, serving as an alternative to XML.

```
{
  "name":"mocha",
  "shop":"costa",
  "rating":3.5,
  "price":2.0,
  "favourite":0,
  "id":1
},
{
  "name":"americano",
  "shop":"costa",
  "rating":4.5,
  "price":3.0,
  "favourite":1,
  "id":2
},
{
  "name":"cappuccino lite",
  "shop":"starbucks",
  "rating":1.5,
  "price":4.0,
  "favourite":1,
  "id":3
}
```

When to use JSON?

- SOAP is a protocol specification for exchanging structured information in the implementation of Web Services.
- SOAP internally uses XML to send data back and forth.

- REST is a design concept.
- You are not limited to picking XML to represent data, you could pick anything really (JSON included).

JSON example

```
{
        "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 vs XML

```
<?xml version="1.0" encoding="UTF-8"?>
<persons>
        <person>
                <firstName>John</firstName>
                <lastName>Smith</lastName>
                <age>25</age>
                <address>
                        <streetAddress>21 2nd Street</streetAddress>
                        <city>New York</city>
                        <state>NY</state>
                        <postalCode>10021</postalCode>
                </address>
                <phoneNumbers>
                        <phoneNumber>
                                <number>212 555-1234</number>
                                <type>home</type>
                        </phoneNumber>
                        <phoneNumber>
                                <number>646 555-4567</number>
                                <type>fax</type>
                        </phoneNumber>
                </phoneNumbers>
        </person>
</persons>
```

JSON vs XML size

- XML: 549 characters, 549 bytes
- JSON: 326 characters, 326 bytes
- **XML ~68,4 % larger than JSON!**

- But a large data set is going to be large regardless of the data format you use.
- Most servers gzip or otherwise compress content before sending it out, the difference between gzipped JSON and gzipped XML isn't nearly as drastic as the difference between standard JSON and XML.

JSON Schema

- Describes your JSON data format
- http://jsonschemalint.com/
- http://json-schema.org/implementations
- http://en.wikipedia.org/wiki/JSON#Schema_and_Metadata

JSON Values

JSON values can be:

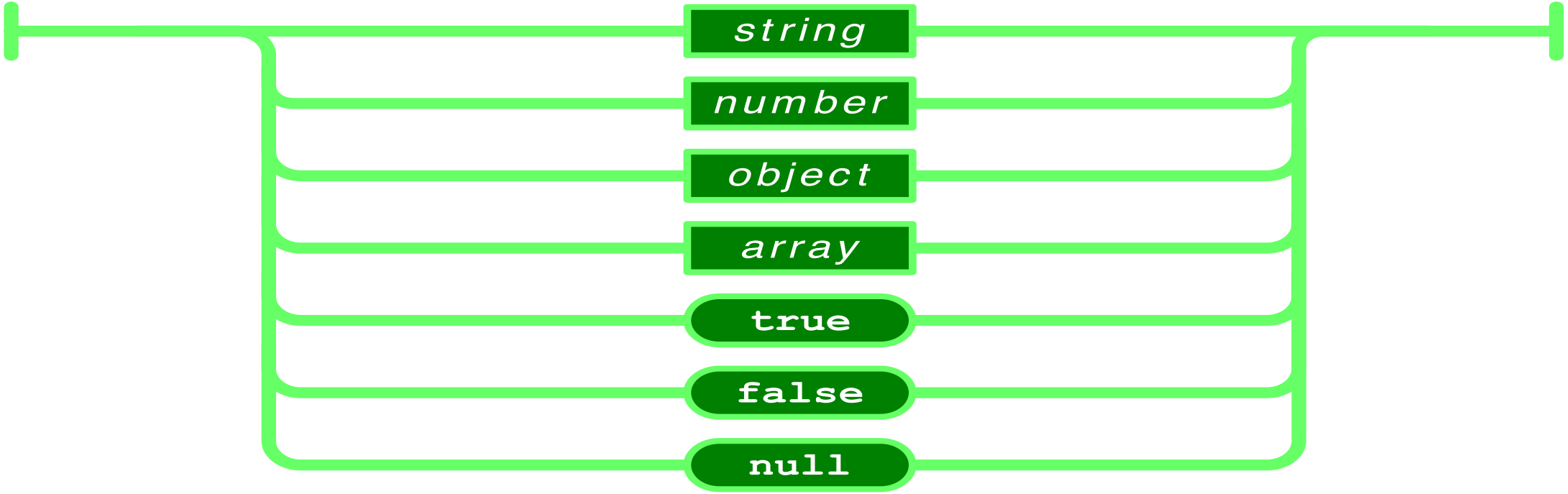
- A number (integer or floating point)
- A string (in double quotes)
- A boolean (true or false)
- An *object* (in curly brackets)
- An *array* (in square brackets)
- null

JSON Values

- Object
  - Unordered set of name-value pairs
  - names must be strings
  - { name1 : value1, name2 : value2, …, nameN : valueN }

- Array
  - Ordered list of values
  - [value1, value2, … valueN]

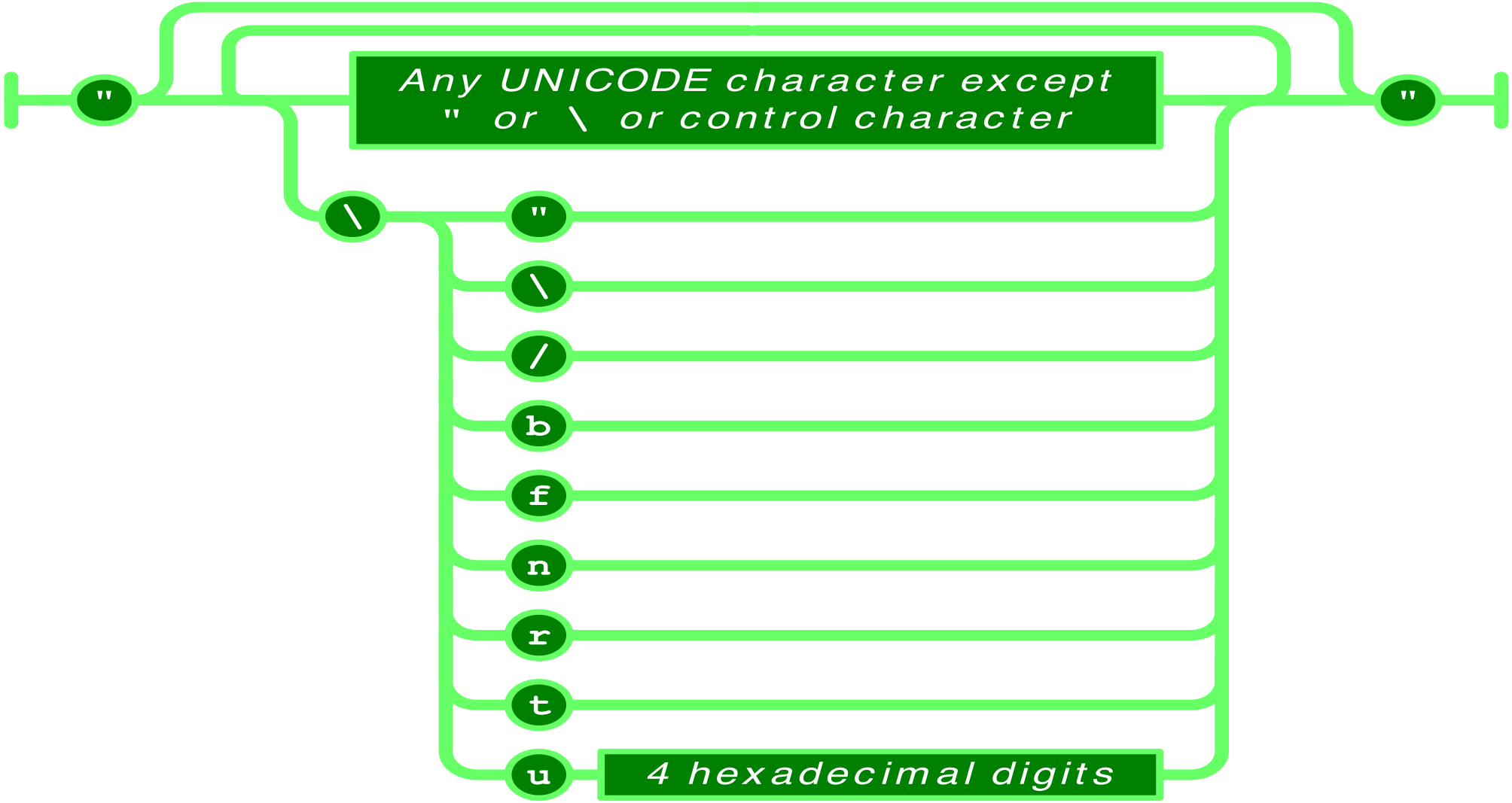
Value



Strings

- Sequence of 0 or more Unicode characters
- No separate character type
  - A character is represented as a string with a length of 1
- Wrapped in "double quotes"
- Backslash escapement

String

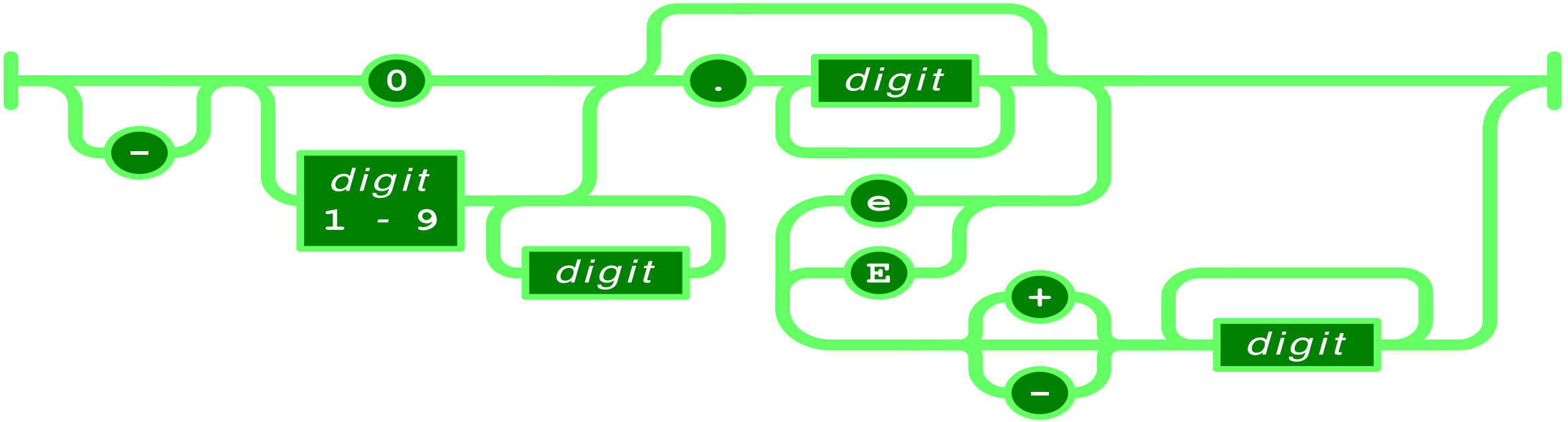

"
Any UNICODE character except
" or \ or control character
"
\
"
\
/
b
f
n
r
t
u
4 hexadecimal digits

Numbers

- Integer
- Real
- Scientific

- No octal or hex
- No **`NaN`** or **`Infinity`**
  - Use **`null`** instead

Number



Booleans

- `true`
- `false`

`null`

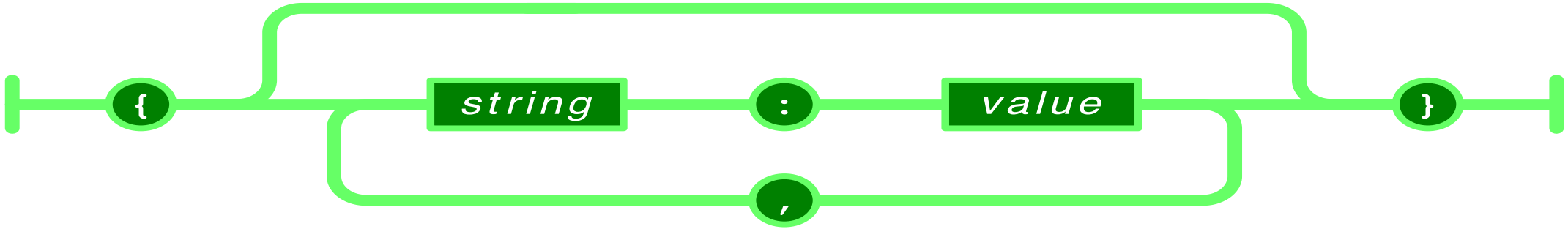
- A value that isn't anything

Object

- Objects are unordered containers of key/value pairs
- Objects are wrapped in `{ }`
- `,` separates key/value pairs
- `:` separates keys and values
- Keys are strings
- Values are JSON values

  - struct, record, hashtable, object

Object

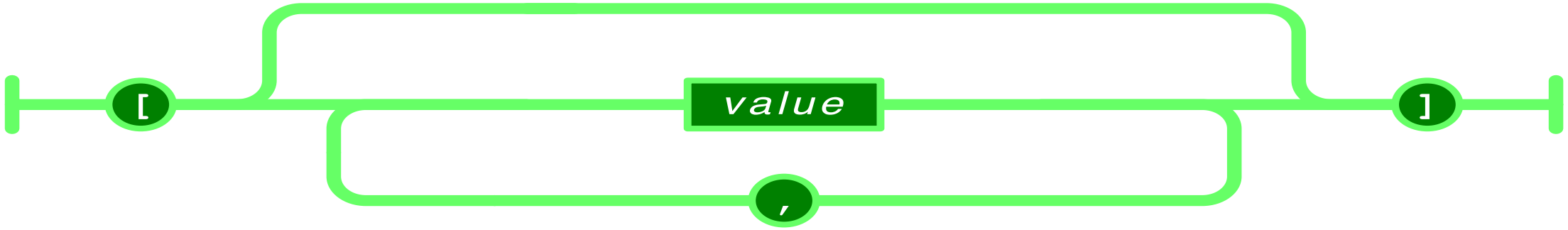



```
{
"_id":"560515770f76130300c69953",
"usertoken":"11343761234567808125",
"paymenttype":"PayPal",
"__v":0,
"upvotes":0,
"amount":1999
}
```

Array

- Arrays are ordered sequences of values
- Arrays are wrapped in `[]`
- `,` separates values
- JSON does not talk about indexing.
  - An implementation can start array indexing at 0 or 1.

Array




```
[
{"_id":"560515770f76130300c69953","usertoken":"11343761234567808125","
paymenttype":"PayPal","__v":0,"upvotes":0,"amount":1999},

{"_id":"561252404218920300048403d","usertoken":"11343761234567808125","
paymenttype":"PayPal","__v":0,"upvotes":5,"amount":1234},

{"_id":"5627620ac9e9e303005b113c","usertoken":"11343761234567808125","
paymenttype":"Direct","__v":0,"upvotes":2,"amount":1001}
]
```

MIME Media Type & Character Encoding

- **`application/json`**
- Strictly UNICODE.
- Default: UTF-8.
- UTF-16 and UTF-32 are allowed.

Versionless

- JSON has no version number.
- No revisions to the JSON grammar are anticipated.
- JSON is very stable.

General Rules

- A JSON decoder must accept all well-formed JSON text.
- A JSON decoder may also accept non-JSON text.
- A JSON encoder must only produce well-formed JSON text.
- *Be conservative in what you do, be liberal in what you accept from others.*

Google’s Gson

https://sites.google.com/site/gson/gson-user-guide

Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson is an open-source project hosted at http://code.google.com/p/google-gson.

Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of.

Example using Google's Gson

- To create a POJO from a JSON String we can do something like this (`.fromJson()`)

```
// Result handling
Coffee result = null;
Type objType = new TypeToken<Coffee>(){}.getType();
result = new Gson().fromJson(response, objType);
```

- To convert a POJO to a JSON String we can do something like this (`.toJson()`)

```
Type objType = new TypeToken<Coffee>(){}.getType();
String json = new Gson().toJson(aCoffee, objType);
```

Sources

- http://en.wikipedia.org/wiki/JSON
- http://www.w3schools.com/json/
- http://www.json.org/
- http://json-schema.org
- http://www.nczonline.net/blog/2008/01/09/is-json-better-than-xml/
- http://en.wikipedia.org/wiki/SOAP_(protocol)
- http://en.wikipedia.org/wiki/REST
- http://stackoverflow.com/questions/16626021/json-rest-soap-wsdl-and-soa-how-do-they-all-link-together

Questions?