Mobile Application Development

Produced by

David Drohan (ddrohan@wit.ie)

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

JSON & Googles Gson

A bit about JSON & Googles Gson

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

- Alternative? - use a standardized system -- JSON
  - Makes the information more portable

JavaScript Object Notation – What is it?

- ❑ Language Independent.
- ❑ Text-based.
- ❑ Light-weight.
- ❑ Easy to parse.

JavaScript Object Notation – What is it?

- JSON is lightweight text-data interchange format
- JSON is language independent*
  - *JSON uses JavaScript syntax for describing data objects
    - JSON parsers and JSON libraries exists for many different programming languages.
- JSON is "self-describing" and easy to understand
- JSON - Evaluates to JavaScript Objects
  - The JSON text format is syntactically identical to the code for creating JavaScript objects.
  - Because of this similarity, instead of using a parser, a JavaScript program can use the built-in eval()*** function and execute JSON data to produce native JavaScript objects.

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 to 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 vs XML

Favor XML over JSON when any of these is true:

- You need message validation
- You're using XSLT
- Your messages include a lot of marked-up text
- You need to interoperate with environments that don't support JSON

Favor JSON over XML when all of these are true:

- Messages don't need to be validated, or validating their deserialization is simple
- You're not transforming messages, or transforming their deserialization is simple
- Your messages are mostly data, not marked-up text
- The messaging endpoints have good JSON tools

Security problems***

- ❑ The eval() function can compile and execute any JavaScript. This represents a potential security problem.
- ❑ It is safer to use a JSON parser (like Gson) to convert a JSON text to a JavaScript object. A JSON parser will recognize only JSON text and will not compile scripts.
- ❑ In browsers that provide native JSON support, JSON parsers are also faster.
- ❑ Native JSON support is included in newer browsers and in the newest ECMAScript (JavaScript) standard.

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:

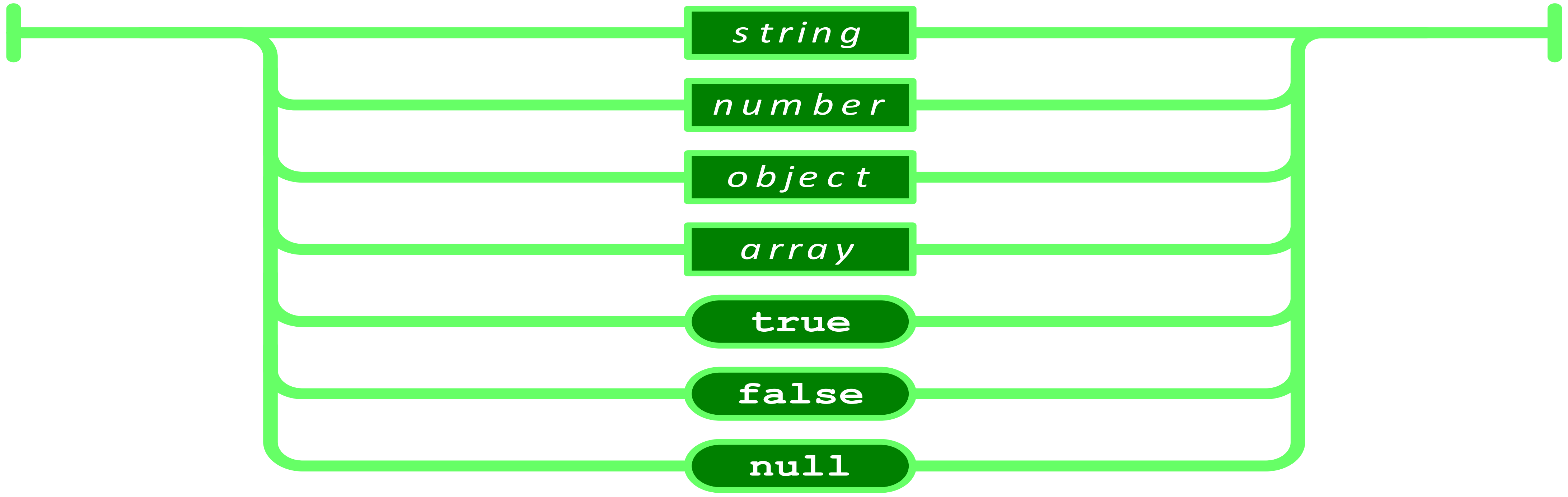
- ❑ A number (integer or floating point)
- ❑ A string (in double quotes)
- ❑ A boolean (true or false)
- ❑ An *object* (in curly brackets)
- ❑ An *array* (in square brackets)
- ❑ null

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]

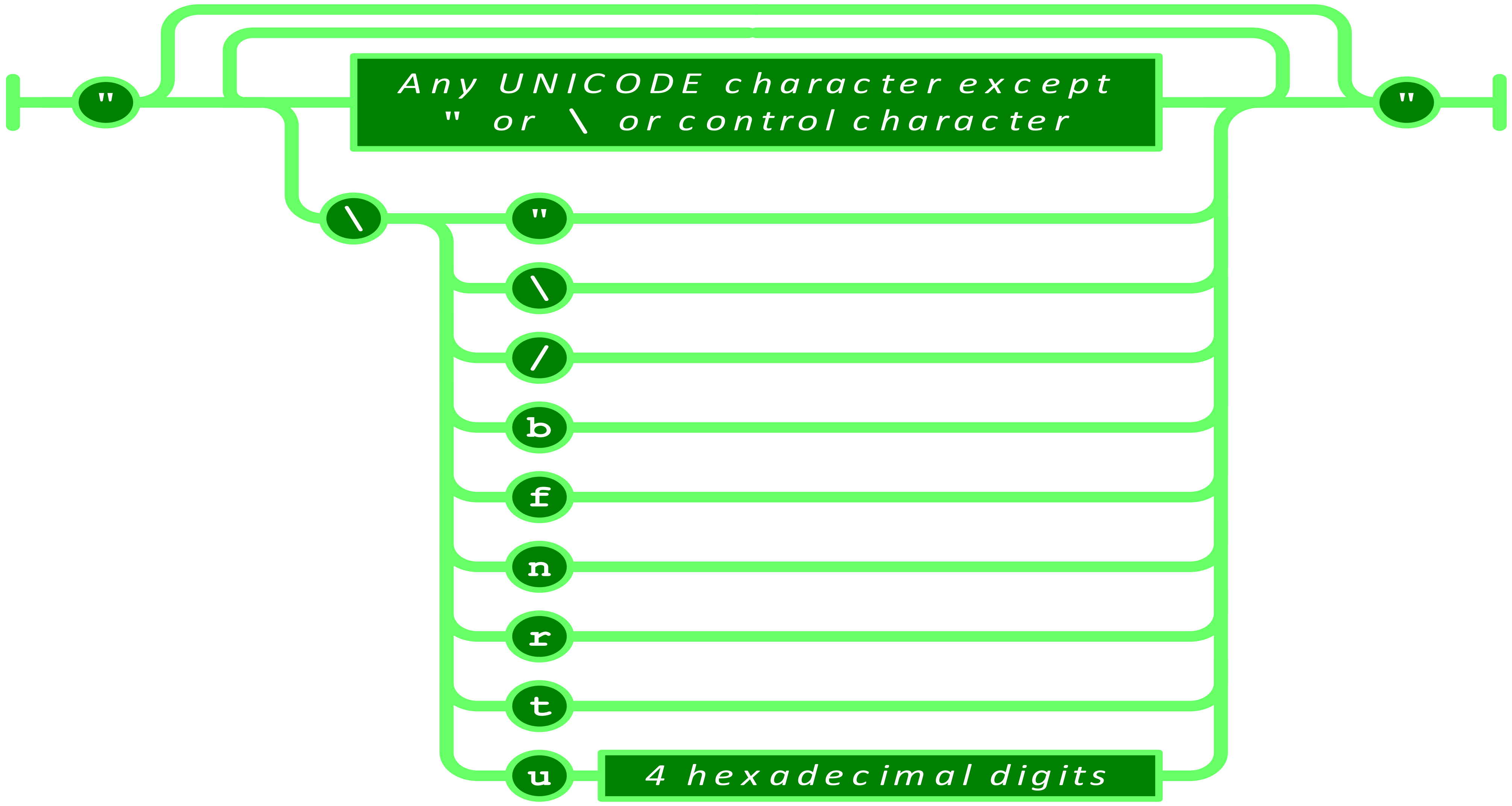
Value



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

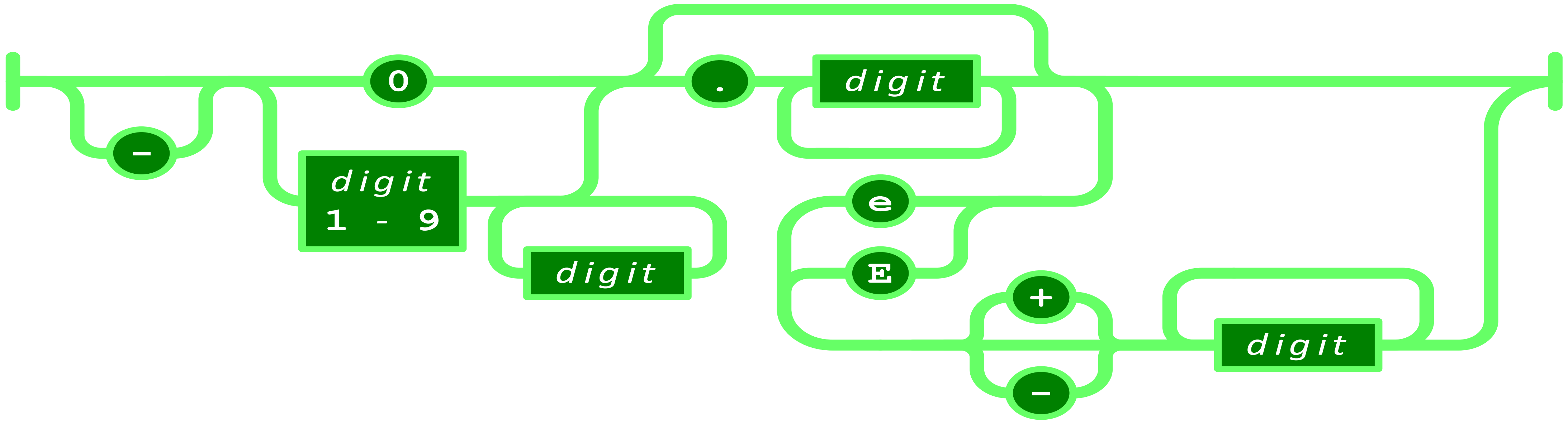

"
Any UNICODE character except " or \ or control character
\
"
\
/
b
f
n
r
t
u
4 hexadecimal digits
"

Numbers

- Integer
- Real
- Scientific

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

Number



Booleans

- `true`
- `false`

`null`

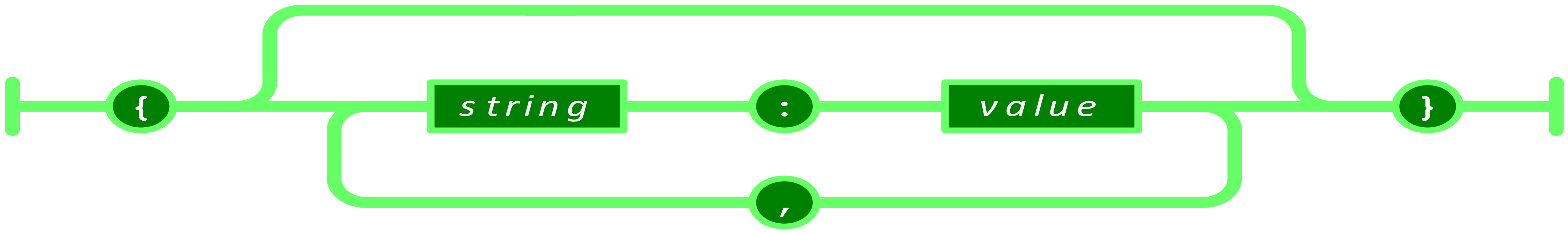
- A value that isn't anything

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

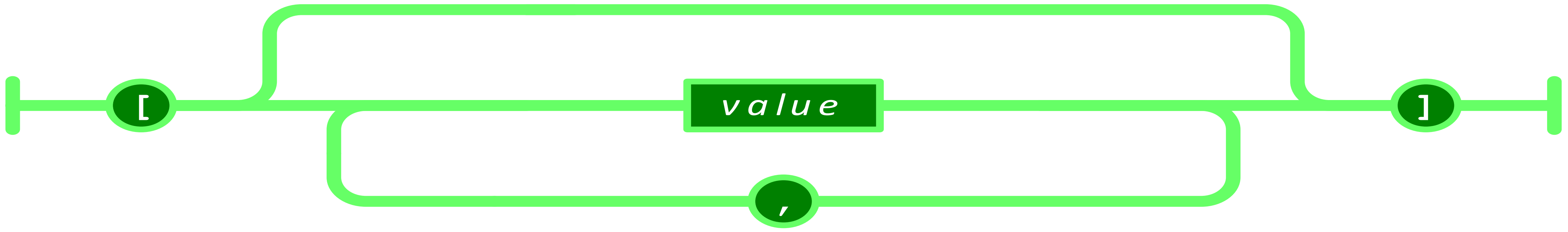


```
{
"_id":"560515770f76130300c69953",
"usertoken":"11343761234567808125",
"paymenttype":"PayPal",
"__v":0,
"upvotes":0,
"amount":1999
}
```

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":"5612524042189203004840 3d","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.

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.*

CoffeeMate
&
Googles Gson

Google’s Gson

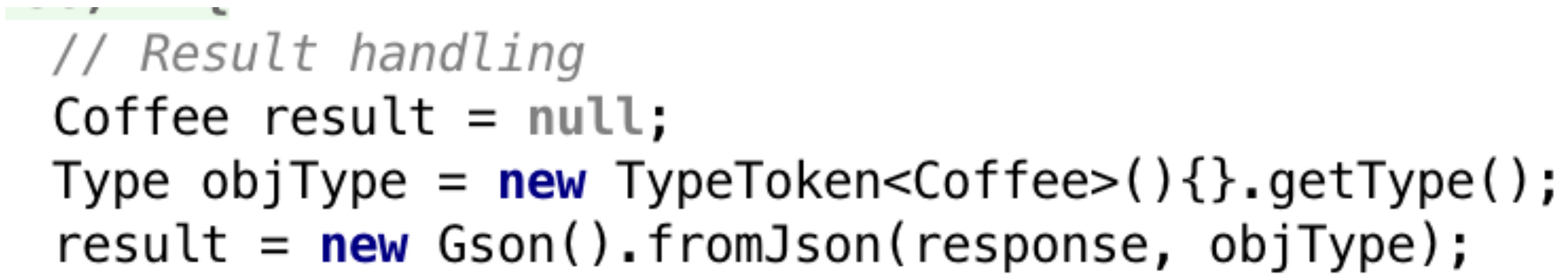
https://sites.google.com/site/gson/gson-user-guide

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.

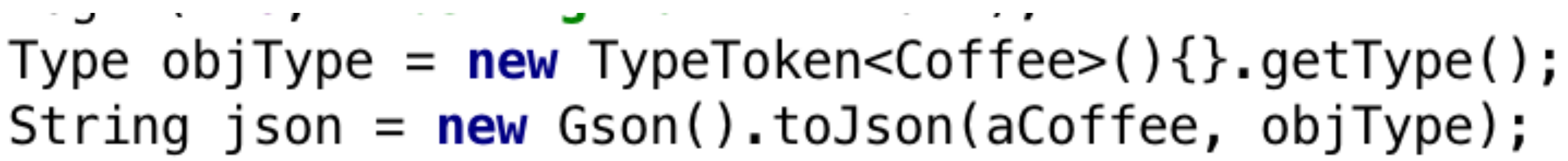
CoffeeMate & Google's Gson

- 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);
```

JSON Summary

- JSON is a standard way to exchange data
  - Easily parsed by machines
  - Human readable form

- JSON uses dictionaries and lists
  - Dictionaries are unordered
  - Lists are ordered

- GSON is Googles JSON parser
  - Very simple to use

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