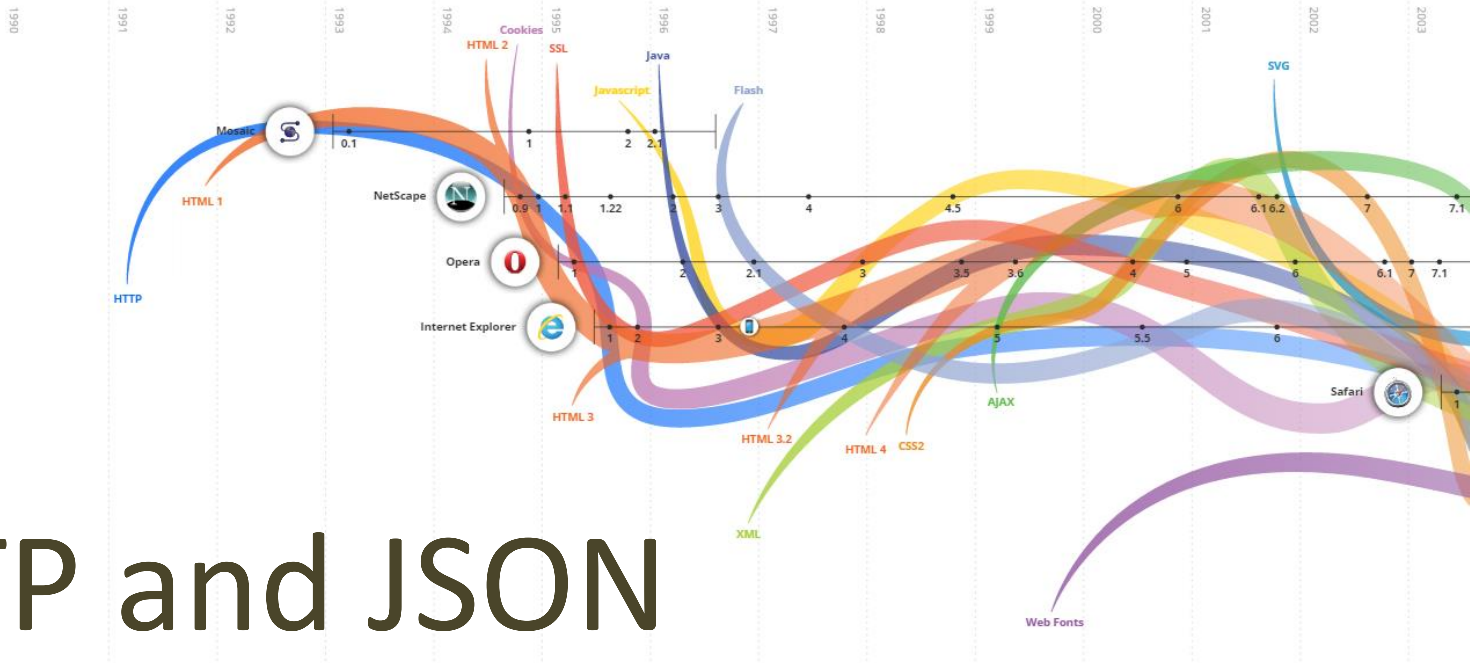


HTTP and JSON

OVERVIEW

The HTTP protocol and the JavaScript Object Notation (JSON) data interchange format

POLITECNICO
DI TORINO

e-Lite

Goal

- Understanding the main communication protocol (HTTP)
- How to represent complex objects to be exchanged over HTTP requests: the JSON data format

Summary

- HTTP (Hypertext Transfer Protocol)
- JSON (JavaScript Object Notation)

Hypertext Transfer Protocol

HTTP

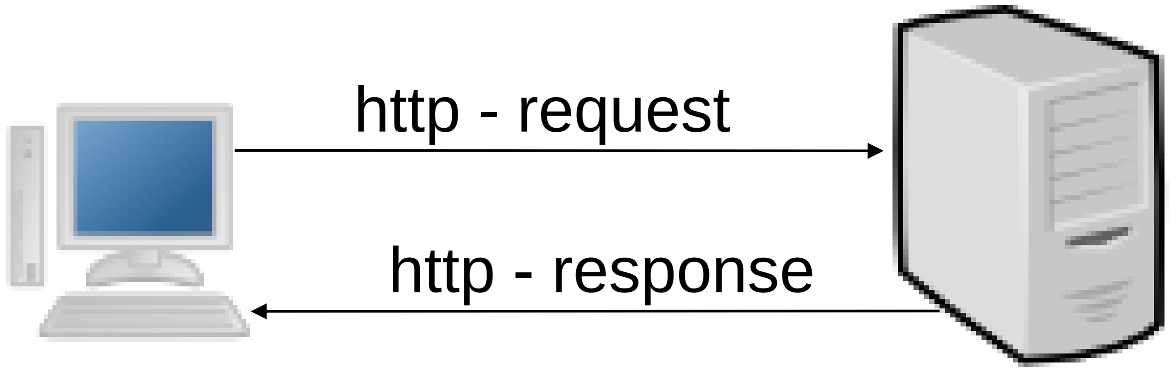
What is HTTP?

- HTTP stands for Hypertext Transfer Protocol
- It is the network protocol used to delivery virtually all data over the WWW:
  - Images
  - HTML files
  - Query results
  - Etc.
- HTTP takes places over TCP/IP connections

http://www.ietf.org/rfc/rfc2616.txt

HTTP clients and servers

- A browser is an HTTP client because it sends requests to an HTTP server, which then sends responses back to the client.
- The standard port for HTTP servers to listen on is 80, though they can use any port.



HTTP messages

- The format of the request and response messages are similar.
  - An initial line
  - Zero or more header lines
  - A blank line (CRLF)
  - An optional message body

```
Initial line
header1: value1
header2: value2
header3: value3

message body...
```

Header Example

```
HEAD /index.html HTTP/1.1
Host: www.example.com
```

Request

Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
```

HTTP request – initial line

- The initial line is different for the request and the response.
- A **request** initial line has three parts separated by white spaces:
  - A method name
  - The local path of the requested resource
  - The version of the HTTP being used

- `GET /path/to/file/index.html HTTP/1.0`

HTTP request – initial line

- The method name is always in upper case.
- There are several methods for a HTTP request
  - GET (most commonly used)
  - POST (used for sending form data)
  - HEAD
  - …
- The path is the part of the URL after the host name
  - http://www.tryme.com/examples/example1.html

HTTP Method Basics

HEAD	Gets just the HTTP header
GET	Gets HTTP head & body
POST	Submits data in the body to the server
PUT	Uploads a resource
DELETE	Deletes a resource
TRACE	Echo’s back the request
OPTIONS	Gets a list of supported methods
CONNECT	Converts to a TCP/IP tunnel for HTTPS
PATCH	Apply partial modifications to a resource

HTTP request – initial line

- The HTTP version is always in the form
  - HTTP/x.x (uppercase)
- The versions currently in use are:
  - HTTP/1.0
  - HTTP/1.1
- HTTP/2 exists
  - standardized in 2015

HTTP response – initial line

- The **response** initial line is usually called status line and has also 3 parts separated by spaces:
  - The HTTP version
  - The response status code
  - An English phrase describing the status code
- Example:
  - `HTTP/1.0 200 OK`
  - `HTTP/1.0 404 Not Found`

Response Status Codes

- 1xx – Informational
- 2xx – Success
- 3xx – Redirection
- 4xx – Client Error
- 5xx – Server Error

Response Status Codes

- 1xx – Informational
- 2xx – Success
- 3xx – Redirection
- 4xx – Client Error
- 5xx – Server Error

- 100 = Continue
- 102 = Processing
- 200 = OK
- 201 = Created
- 204 = No Content
- 206 = Partial Content
- 301 = Moved Permanently
- 302 = Found (Moved Temp)
- 307 = Temp Redirect
- 400 = Bad Request
- 401 = Unauthorised
- 402 = Payment Required
- 403 = Forbidden
- 404 = Not Found
- 405 = Method Not Allowed
- 409 = Conflict
- 450 = Blocked by Windows Parental Controls
- 500 = Internal Server Error
- 501 = Not Implemented

HTTP msg – header lines

- Header lines provide information about the request/response or about the object sent in the message body
- The header lines are in the following format:
  - One line per header
  - Form: “Header-Name: value”
- HTTP/1.0 defines 16 headers (none required); HTTP/1.1 defines 46 headers and 1 is required in requests:
  - `Host:`

Request headers

- Accept
- Accept-Charset
- Accept-Encoding
- Accept-Language
- Authorization;
- Expect
- From
- **Host**
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Proxy-Authorization
- Range
- Referer
- TE
- User-Agent

Response Headers

- `Accept-Ranges`
- `Age`
- `Etag`
- `Location`
- `Proxy-Authenticate`
- `Retry-After`
- `Server`
- `Vary`
- `WWW-Authenticate`

General (request & response) headers

- `Cache-Control`
- `Connection`
- `Date`
- `Pragma`
- `Trailer`
- `Transfer-Encoding`
- `Upgrade`
- `Via`
- `Warning`

Message body

- An HTTP message may have a **body** of data sent after the header lines.
- In a **response** the body contains the resource returned to the client
  - Images
  - text/plain, text/html
  - …
- In a **request** it may contain the data entered by the user in a form or a file to upload, etc.

Content Type

- Proper name: Internet Media Type
  - Also known as MIME type
- Parts: Type, SubType, Optional Parameters
- `x-` prefix for nonstandard types or subtypes
- `vnd.` prefix for vendor specific subtypes

Content Type Examples

Content-Type	File
text/plain	Plain text
text/xml	XML
text/html	HTML
image/png	PNG image
audio/basic	Wave audio
audio/mpeg	MPEG audio (MP3)
video/quicktime	Quicktime Video
application/pdf	Adobe PDF document
application/javascript	JavaScript
application/vnd.ms-powerpoint	PowerPoint file
application/json	JSON

Message body

- Some HTTP headers are used to describe the body content:
  - `Allow`
  - `Content-Encoding`
  - `Content-Language`
  - `Content-Length`
  - `Content-Location`
  - `Content-MD5`
  - `Content-Range`
  - `Content-Type`
  - `Expires`
  - `Last-Modified`
  - `extension-header n`

HTTP Authentication

- Basic Authentication
  - Easy to do, but plain text. Easy to reverse engineer. Less of an issue when used with SSL.
- Digest Authentication
  - Harder to do, still plain text. Hard (impossible?) to reverse engineer because of hashing.
- NTLM Authentication
  - Hard to do, Windows specific. Hard (impossible?) to reverse engineer.
- Note: usually, authentication is dealt at the application level, and http mechanisms are not used

HTTP methods: HEAD

- The HEAD method is like the GET except it asks the server to return the **response headers, only**. Is useful for checking the characteristics of a resource without actually downloading it.
- The response to a HEAD request **never** contains a message body, only the initial line and the headers.

HTTP methods: POST

- Used to send data to the server
- A POST request is different from the GET request as:
  - There's a block of data sent with the request in the request message body
  - The request URI is not a resource to retrieve, it's usually a program or a server page that handles the sent data
  - The HTTP response is usually not-static (generated depending on the received data)

GET vs POST

- The most common use of the POST method is to submit data gathered from user forms
- Also the GET can be used to submit form data however, the data is encoded in the request URI
  - http://www.example.com/example.html?var=This+is+a+simple+%26+short+test
- GET requests should be **idempotent**, i.e., may be repeated without changing the state of the application

HTTP as transport layer

- HTTP is used as “transport” for many resources / protocols
- Protocols:
  - SOAP (Simple Object Access Protocol)
  - XML-RPC
  - WebDAV
- Resources:
  - Text (plain, HTML, XHTML, ...)
  - Images (gif, jpeg, ...)
  - ....

Formal HTTP standard

- HTTP
  - http://www.w3.org/Protocols/
  - Hypertext Transfer Protocol -- HTTP/1.1: http://tools.ietf.org/html/rfc2616

JavaScript Object Notation

JSON

JSON – What is it?

- "JSON (JavaScript Object Notation) is a lightweight data interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate"

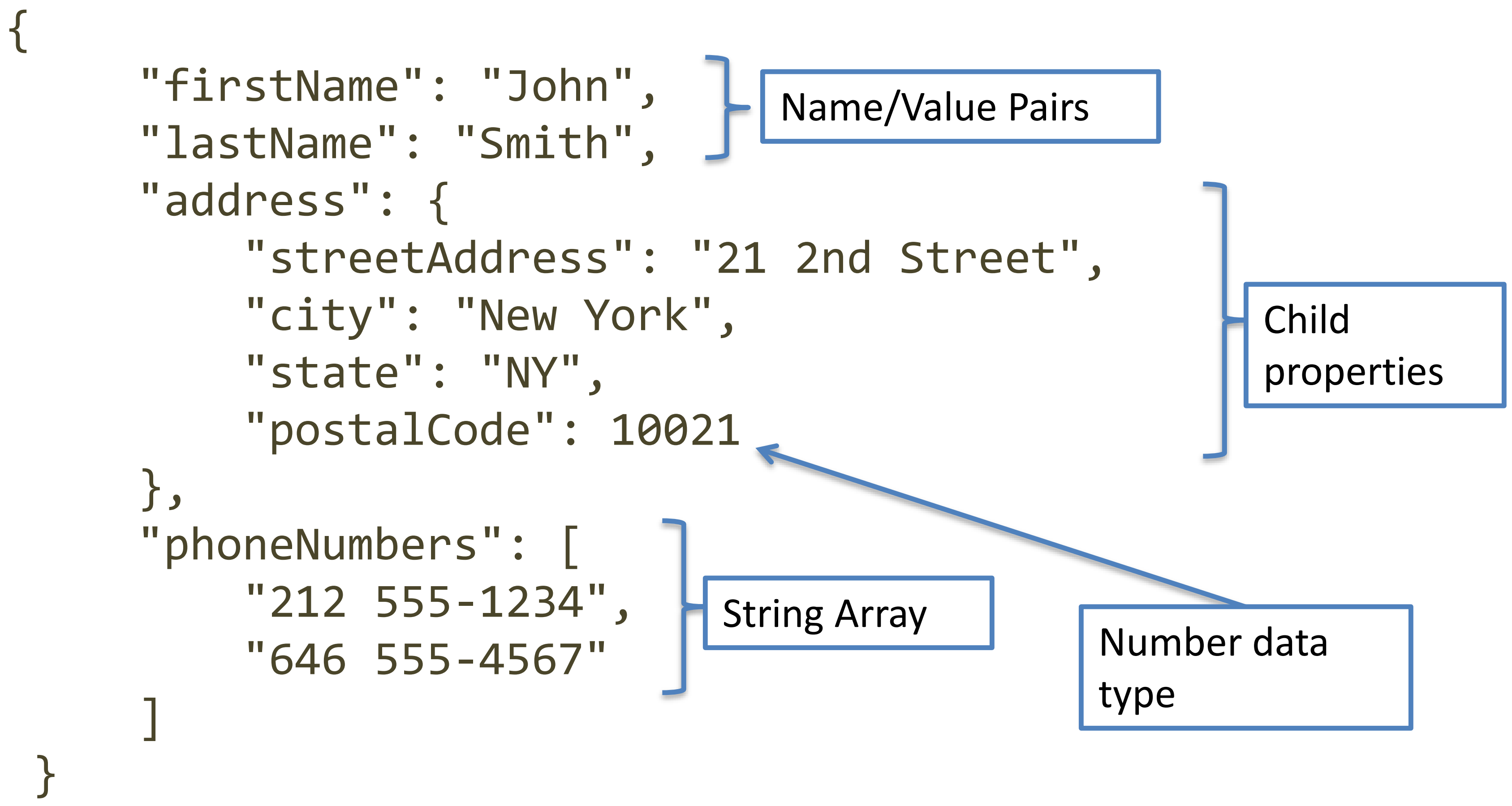
  - *JSON.org*

- Important:
  - JSON is a <u>subset</u> of JavaScript

JSON Logical Structure

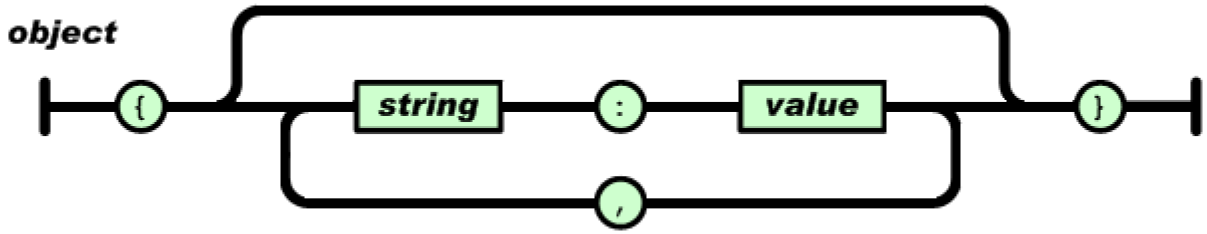
- JSON is built on two structures:
  - A **collection** of name/value pairs. In various languages, this is realized as an ***object***, record, struct, dictionary, hash table, keyed list, or associative array. **{ … }**
  - An **ordered list** of values. In most languages, this is realized as an ***array***, vector, list, or sequence.
    […]

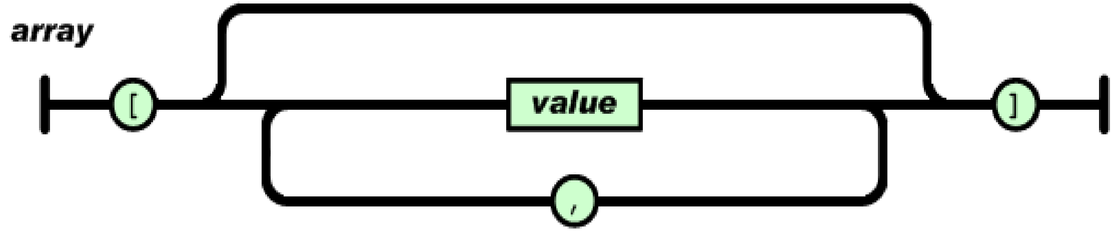
JSON – What does it look like?

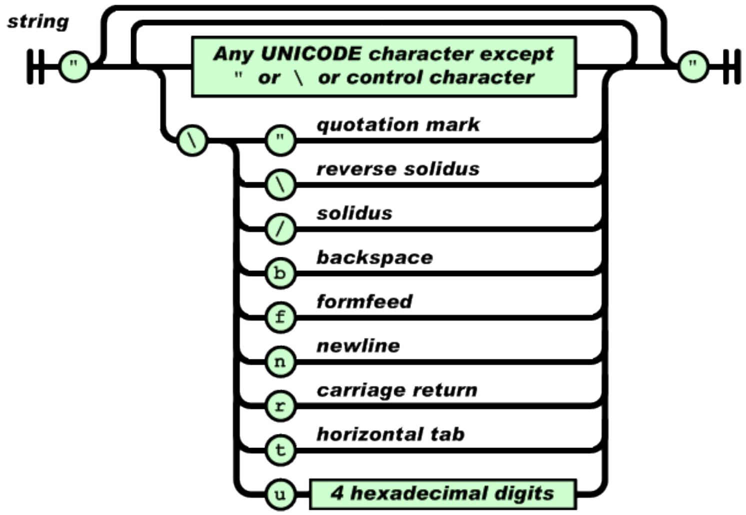


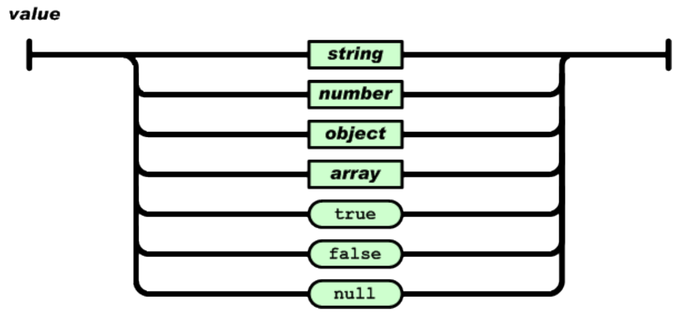
```
{
    "firstName": "John",
    "lastName": "Smith",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        "212 555-1234",
        "646 555-4567"
    ]
 }
```

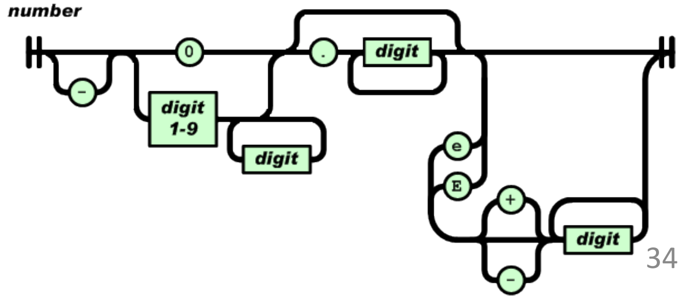
JSON Data Structures











Resources

- JSON
  - http://json.org
  - ECMA-404 The JSON Data Interchange Standard. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
- HTTP
  - http://www.w3.org/Protocols/
  - Hypertext Transfer Protocol -- HTTP/1.1: http://tools.ietf.org/html/rfc2616

License