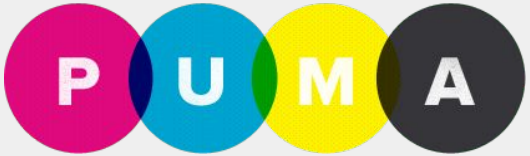
Let’s build a simple HTTP server

RubyConf 2019

WEBrick

thin

Unicorn



Passenger

LEXOO

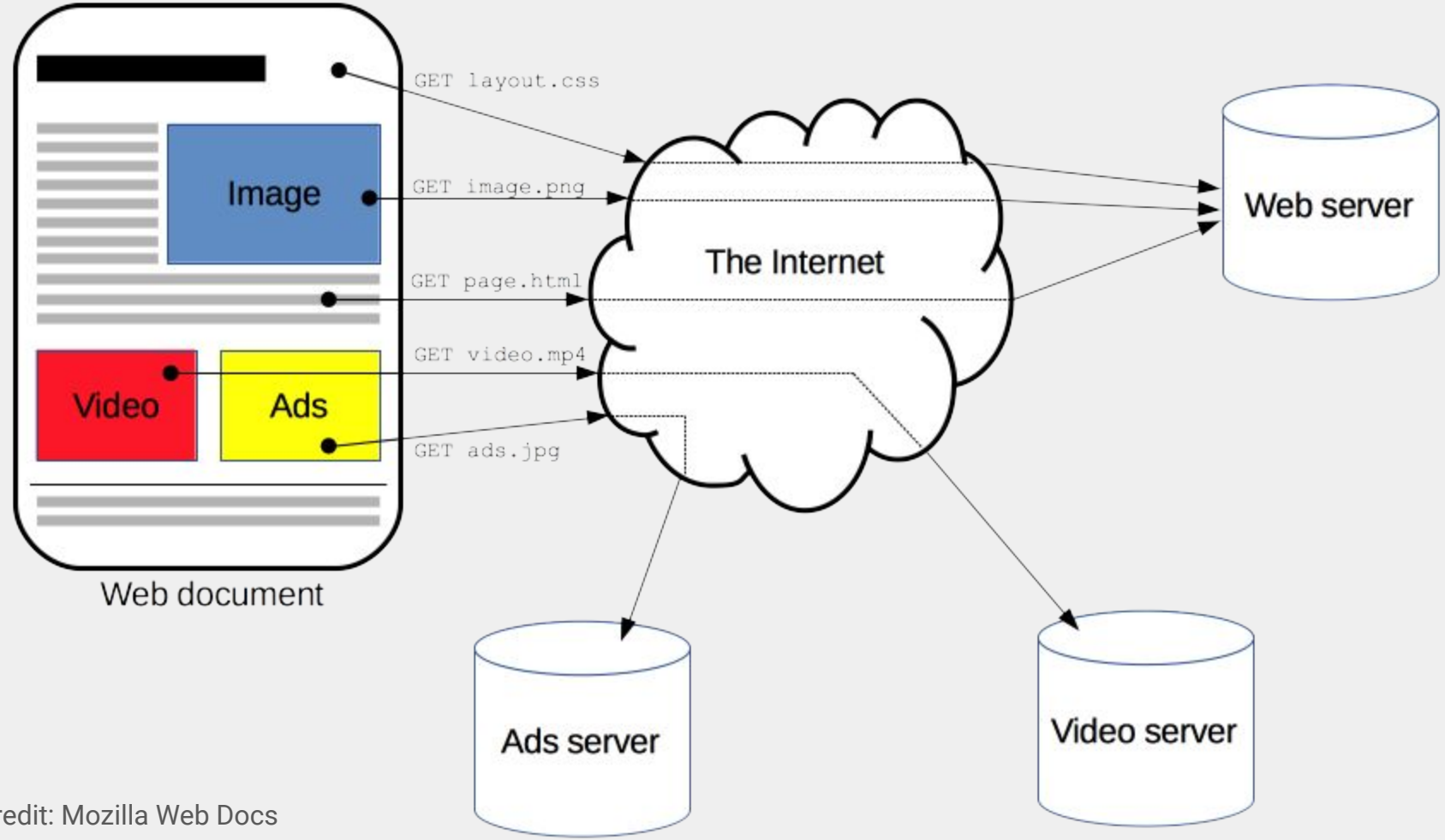
Esther Olatunde

I write code that (sometimes) works on computers

🇳🇬Lagos, Nigeria

What happens when you visit a URL in a browser?

How does the Internet work?



Credit: Mozilla Web Docs

Internet Engineering Task Force (IETF)
Request for Comments: 7230
Obsoletes: 2145, 2616
Updates: 2817, 2818
Category: Standards Track
ISSN: 2070-1721

R. Fielding, Editor
Adobe
J. Reschke, Editor
greenbytes
June 2014

Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing

Abstract

The Hypertext Transfer Protocol (HTTP) is a stateless application-level protocol for distributed, collaborative, hypertext information systems. This document provides an overview of HTTP architecture and its associated terminology, defines the "http" and "https" Uniform Resource Identifier (URI) schemes, defines the HTTP/1.1 message syntax and parsing requirements, and describes related security concerns for implementations.

Status of This Memo

PROPOSED STANDARD
Updated by: 8615
This document has errata.

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7230.

RFC 7230

https://httpwg.org/specs/

Uniform Resource Locator (URL) syntax

```
protocol://hostname:port/path-or-file-name
```

When you visit this URL

```
http://www.example.com/lagos.html
```

HTTP Request

```
GET /lagos.html HTTP/1.1

Host: www.example.com
User-Agent: curl/7.54.0
Accept: */*
```

HTTP Request

```
GET /lagos.html HTTP/1.1

Host: example.com
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 11_0 like Mac OS X)..
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,...
Accept-Encoding: gzip, deflate
Accept-Language: en-US
```

HTTP Response

```
HTTP/1.1 404 Not Found

Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Wed, 20 Nov 2019 09:59:34 GMT
Expires: Wed, 27 Nov 2019 09:59:34 GMT
Last-Modified: Wed, 13 Nov 2019 11:44:22 GMT
Server: ECS (ord/5726)
Vary: Accept-Encoding
X-Cache: 404-HIT
Content-Length: 1256

<!doctype html>
<html>
...
</html>
```

What do we need?

```
require "socket"
```

Socket documentation:

https://ruby-doc.org/stdlib-2.6.5/libdoc/socket/rdoc/index.html

1. Listen for connections
2. Parse the request
3. Build and send the response

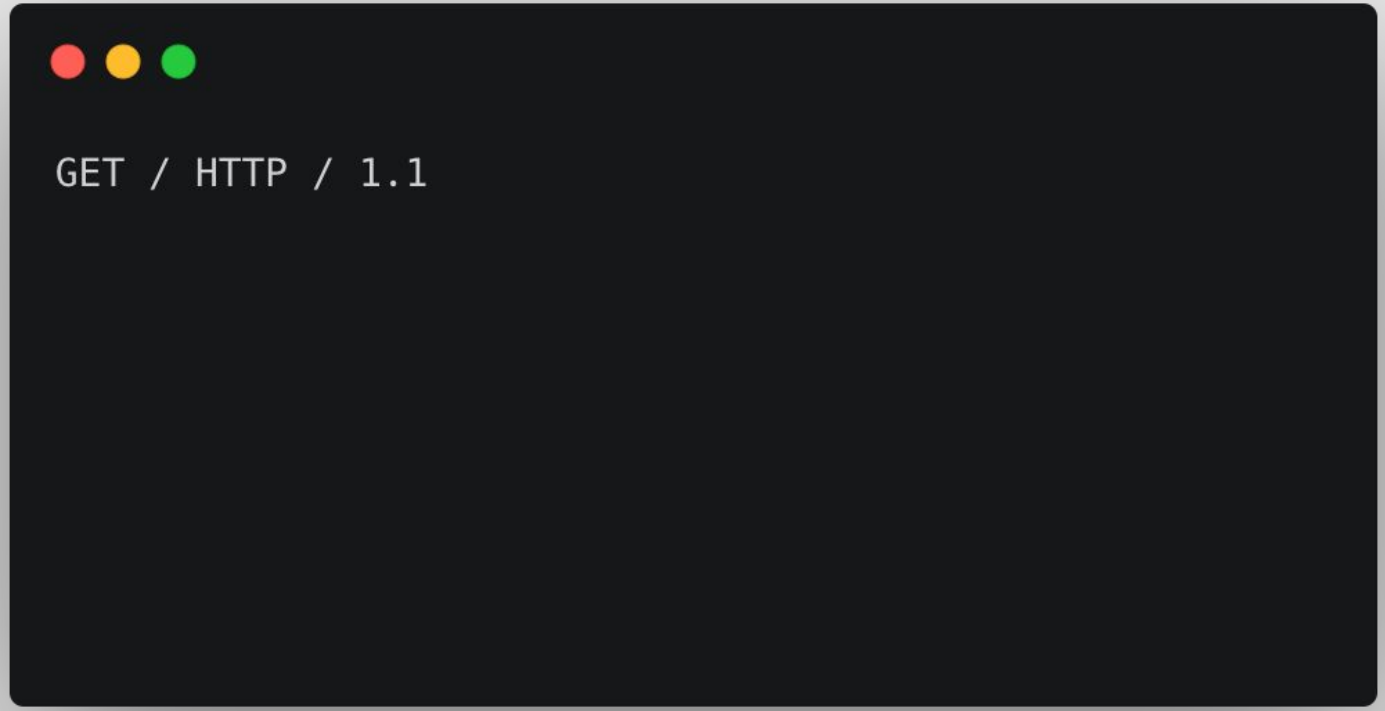
1. Listen for connections

```
require "socket"

server = TCPServer.new("localhost", 5000)

loop do
  socket = server.accept
  request = socket.gets

  STDERR.puts request
end
```


GET / HTTP / 1.1

```
require "socket"

server = TCPServer.new("localhost", 5000)

loop do
  client = server.accept
  request = socket.readpartial(2048)

  STDERR.puts request
end
```


GET / HTTP/1.1
Host: localhost:5000
User-Agent: curl/7.54.0
Accept: */*

2. Parsing the HTTP request

```
GET / HTTP/1.1
Host: localhost:5000
User-Agent: curl/7.54.0
Accept: */*
```

```
def parse(request_string)
  method, path, version = request_string.lines[0].split

  {
    method: method,
    path: path,
    headers: parse_headers(request_string)
  }
end
```

```ruby
def parse_headers(request)
  headers = {}
  request.lines[1..-1].each do |line|
    return headers if line == "\r\n"
    header, value = line.split
    header        = normalize(header)
    headers[header] = value
  end
end


def normalize(header)
  header.gsub(":", "").downcase.to_sym
end
```

```
{
  :method=>"GET",
  :path=>"/",
  :headers=>
    {:host=>"localhost:5000",
     :"user-agent"=>"curl/7.54.0",
     :accept=>"*/*"
     }
}
```

3. Build and send the HTTP response

```
def prepare
  path = parsed_request.fetch(:path)
  if path == "/"
    respond_with(SERVER_ROOT_PATH + "index.html")
  else
    respond_with(SERVER_ROOT_PATH + path)
  end
end
```

```
def respond_with(path)
  if File.exists?(path)
    ok_response(File.binread(path))
  else
    not_found_response
  end
end
```

```ruby
def ok_response(body)
  Lagos::Response.new(code: 200, body: body)
end

def not_found_response
  Lagos::Response.new(code: 404)
end
```

```ruby
class Lagos::Response
  def initialize(code:, body: "")
    @response =
    "HTTP/1.1 #{code}\r\n" +
    "Content-Length: #{body.size}\r\n" +
    "\r\n" +
    "#{body}\r\n"
  end

  def send(client)
    client.write(@response)
  end
end
```

```ruby
require "socket"
require_relative "http_request_parser"
require_relative "http_response_parser"

server = TCPServer.new("localhost", 5000)

loop do
    client = server.accept
    request = client.readpartial(2048)

    request = Lagos::HttpRequest.new(request).parse

    STDERR.puts request

    response = Lagos::HttpResponse.new(request).parse

    response.send(client)

    client.close
end
```

More features

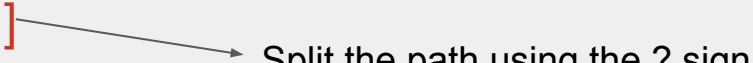
- Execute a ruby file
- Parse a query

Executing a ruby file

```
if File.executable?(path)
    `#{path}`
else
    File.binread(path)
end
```

Parse a query

```
def parse
    method, path, version = @request.lines[0].split
    query = path.split('?')[1]
 {
     method: method,
     path: path,
     query: query,
     headers: parse_headers(@request)
 }
end
```



Is our http server ready to be made available on the internet?

No!

Path traversal attack (also known as directory traversal) aims to access files and directories that are stored outside the web root folder

```
http://localhost:5000/../../../../some
dir/somefilea
```

Where does Rack fit in?

- Rack is the HTTP interface for Ruby
- Rack defines a standard interface for interacting with HTTP and connecting web servers.

Further reading and resources

HTTP Spec:
https://httpwg.org/specs/rfc7540.html

Socket Doc:
https://ruby-doc.org/stdlib-2.6.5/libdoc/socket/rdoc/index.html

Rack Spec:
https://www.rubydoc.info/github/rack/rack/file/SPEC

Explore what's underneath the web servers you use:

https://docs.ruby-lang.org/en/2.4.0/WEBrick.html

https://puma.io/puma/

https://bogomips.org/unicorn/

https://github.com/macournoyer/thin

https://www.phusionpassenger.com/library/walkthroughs/basics/ruby/fundamental_concepts.html

Thank you!

Where to find me on the internet

- @MsEOlatunde
- https://estherolatunde.com