HTTP

Overview

- URLs
- HTTP
- Requests / Responses
- HTTP Verbs
- Authentication / Caching (super basic intro)

URL: Definition

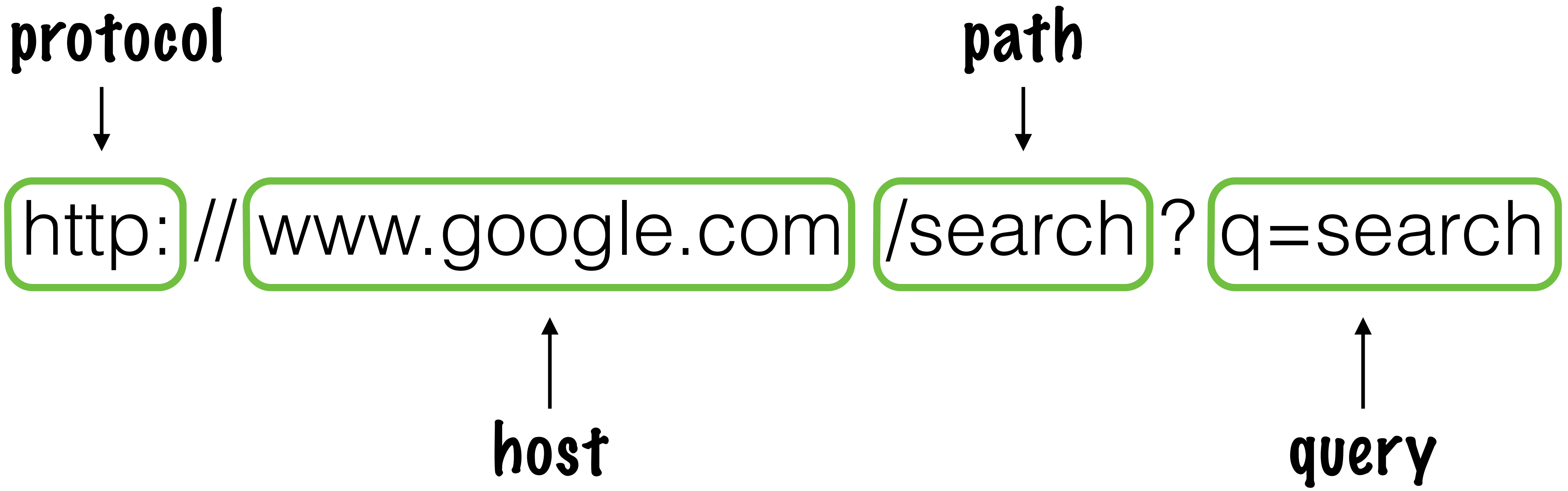
A string of characters that describes the location of a resource on a network.

URL: Parts

Protocol -	*http:, ftp:, https:*
//	
Host -	*galvanize.it*
Port -	*80, or 443 (default is 80)*
Path -	*/products/35*
Query -	*?q=ruby&f=pdf*

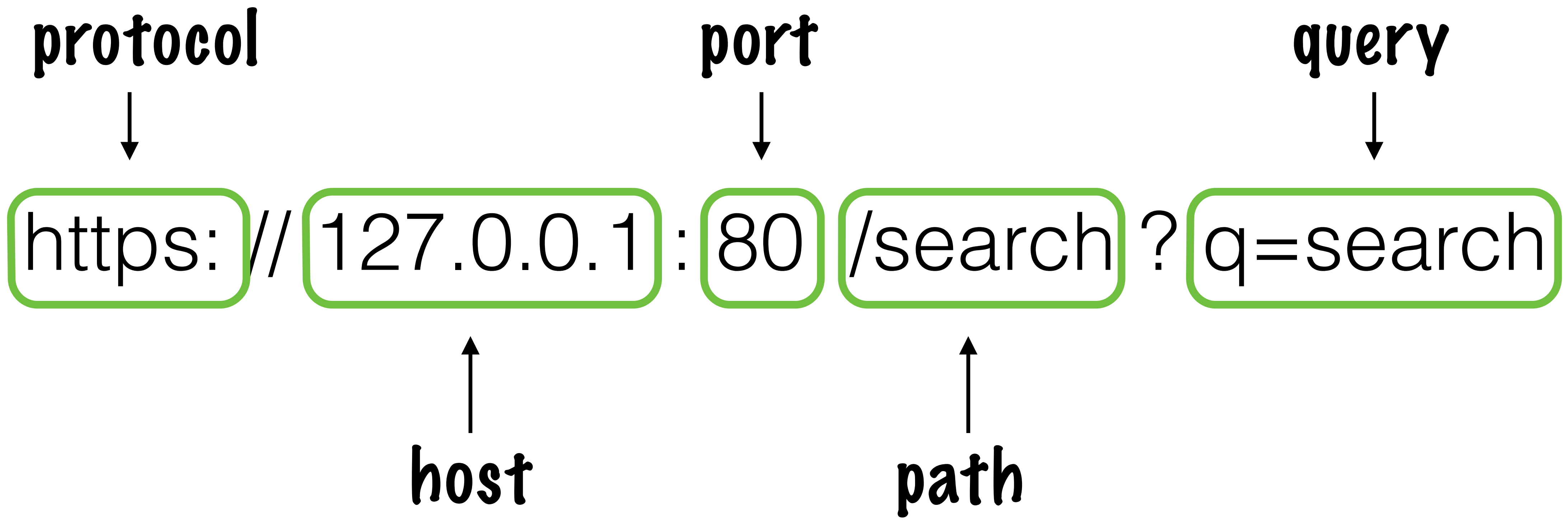
URL: Format

"http:" "//" host [":" port] [path ["?" query]]



URL: Format

"http:" "//" host [":" port] [path ["?" query]]



URL Encoding

Certain characters have special meaning in URLs. When your URL needs to use those characters, you need to encode them. For example:

I look for []

↓

I%20look%20for%20%5B%5D

HTTP: Definition

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.[1]

[1] http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

HTTP: Definition

HT	Hypertext
T	Transfer
P	Protocol

HTTP: Definition

$\boxed{\textbf{HT} \text{ Hypertext}} \longrightarrow$ **HTML (aka markup)**

```
<p>
 This is html
</p>
```

T Transfer

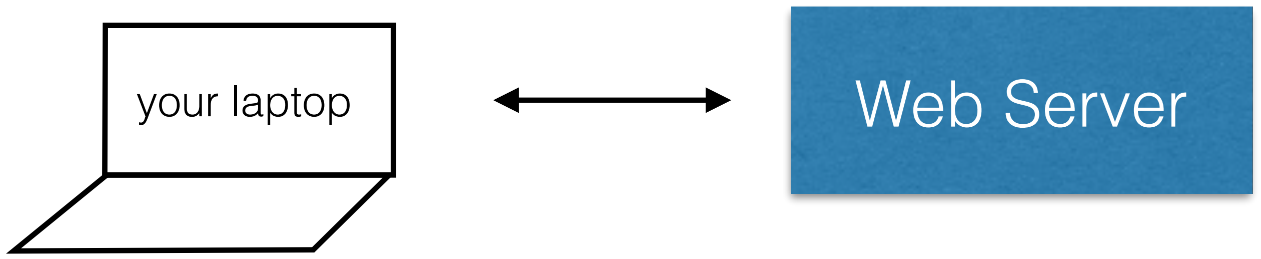
P Protocol

HTTP: Definition

HT Hypertext

T Transfer → **from computer to computer**

P Protocol



HTTP: Definition

HT Hypertext

T Transfer

P Protocol → a set of rules about the format of the text snippets

```
GET / HTTP/1.1
User-Agent: curl/7.30.0
Host: www.google.com
Accept: */*
```

HTTP: Definition

A set of rules about

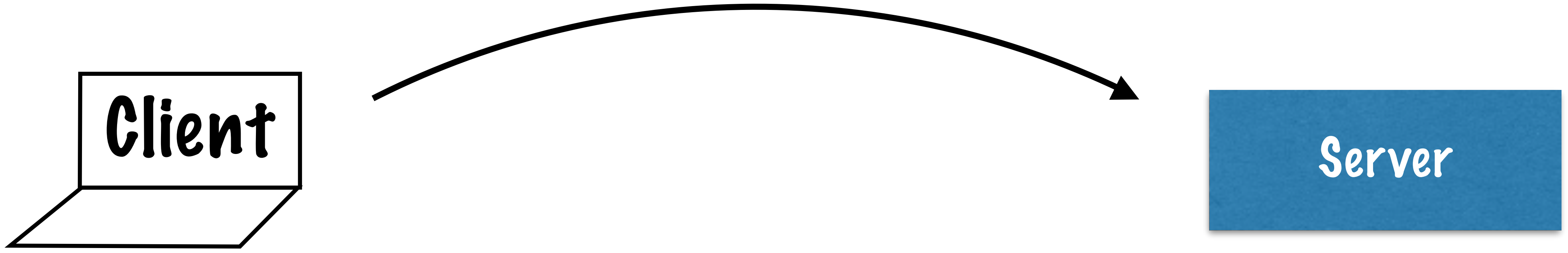
how to ask other computers for information,

and send information to those computers

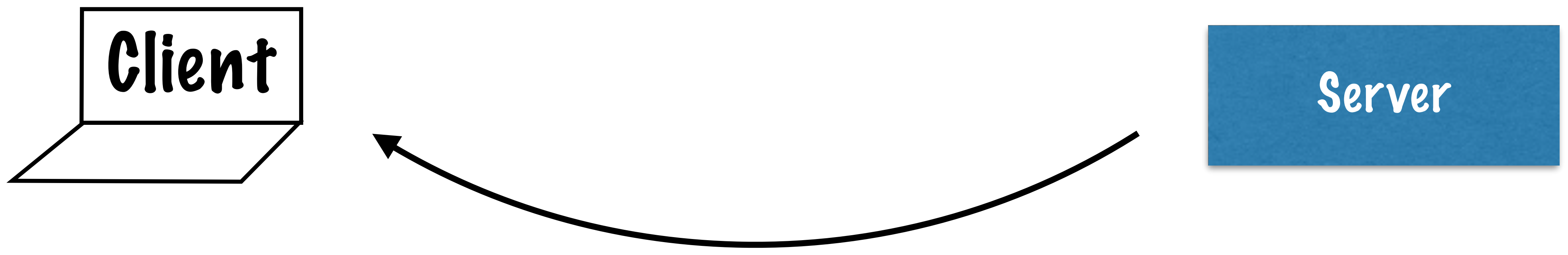
Request / Response

```
GET / HTTP/1.1
User-Agent: curl/7.30.0
Host: www.google.com
Accept: */*
```

the client (your browser) sends a request



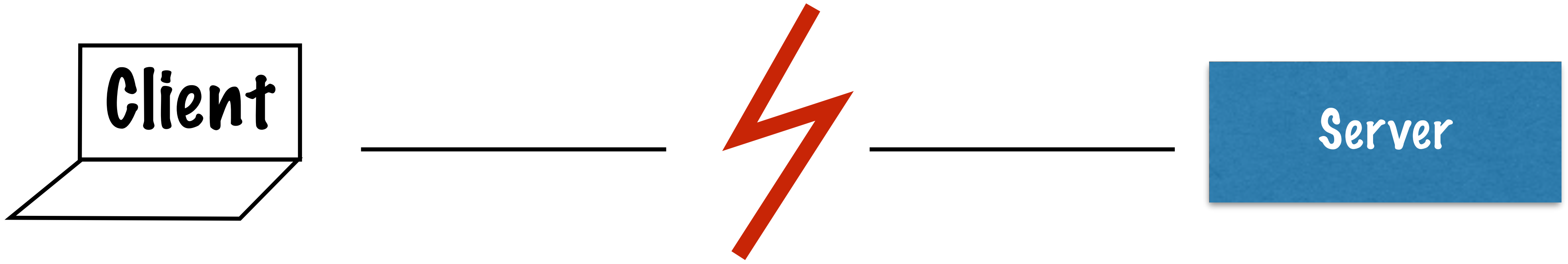
Request / Response



the server (like google.com) returns a response

```
HTTP/1.1 200 OK
Date: Sun, 16 Mar 2014 06:09:13 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: NID=67=euKRU; expires=Mon
Transfer-Encoding: chunked

<html>
  <head></head>
</html
```

Request / Response



As soon as the response has been sent,
the client and server effectively
stop communicating with each other
until the next request is sent.

This is referred to as being“stateless”.

HTTP Requests

Request: Definition

An HTTP request is a snippet of text

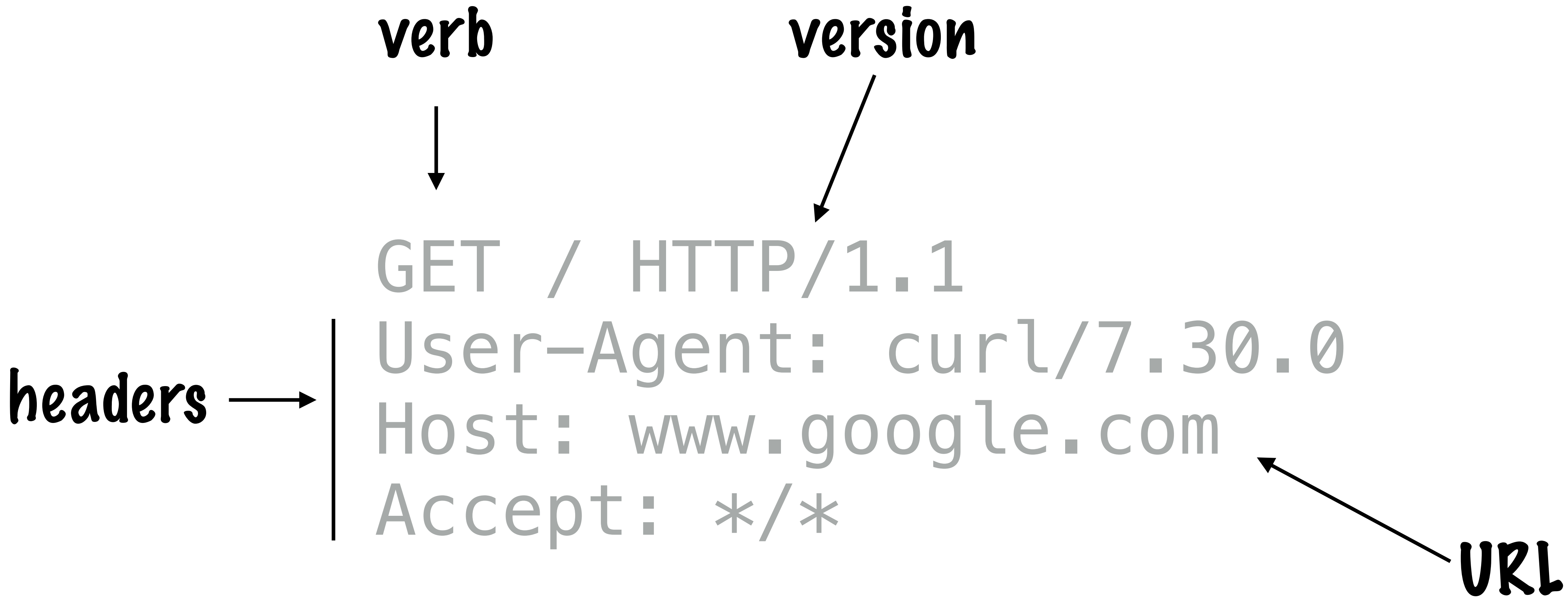
that gets sent to a URL,

and includes an HTTP version,

a valid HTTP verb,

headers and an optional request body

Request: Example



```
GET / HTTP/1.1
User-Agent: curl/7.30.0
Host: www.google.com
Accept: */*
```

Request: Analogy

Imagine you work at an office, and there is a clerk who manages all of the files.

And HTTP request is like asking the clerk:

“Hey, can you get me the Peterson file?”

or *“Here, can you file this form for me?”…*

Request: Analogy

… except that the clerk needs to

be asked in a very particular way.

HTTP Request Verbs

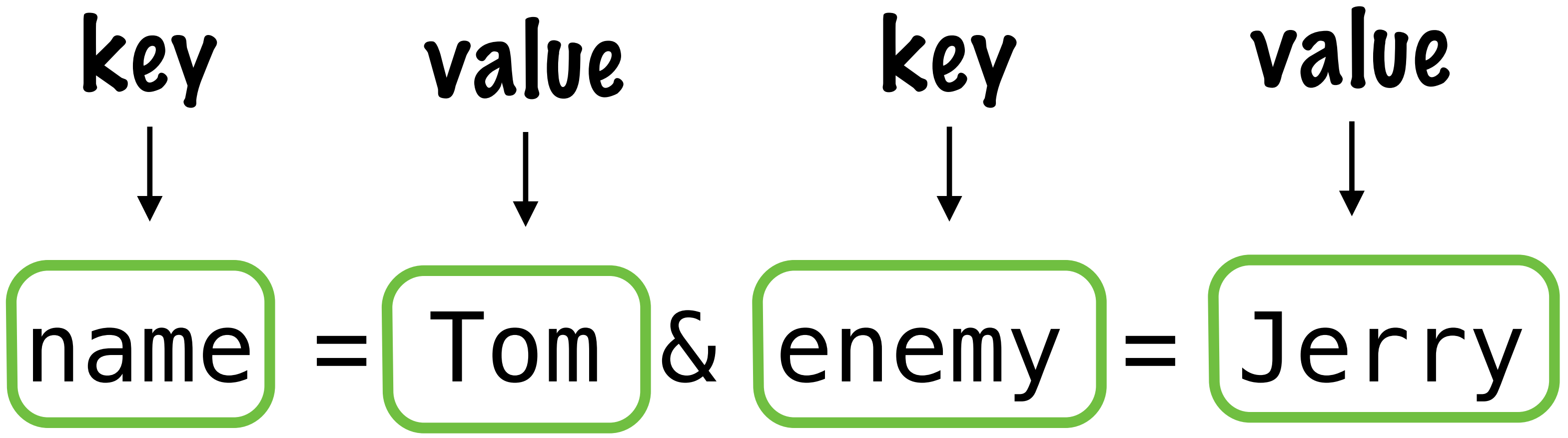
You can think about HTTP verbs in terms of web pages:

- GET - ask for a page on the server
- POST - tell the server to add a new page
- PUT - tell the server to replace an existing page
- PATCH - tell the server to update an existing page
- DELETE - tell the server to delete a page

HTTP Request Body

HTTP Requests can have a body. A typical POST request from a browser adds a request body in the url query string format:

```
name=Tom&enemy=Jerry
```



HTTP Request Body

```
POST /accounts/ClientLogin HTTP/1.1
User-Agent: curl/7.30.0
Host: www.google.com
Accept: */*
Content-Length: 112
Content-Type: application/x-www-form-urlencoded

Email=joe%40example.com&Passwd=new%2Bfoundland
```

↑

request body as url-encoded query string

Request Body Analogy

When you send a url-encoded query string in the request body, it’s like writing down some instructions and giving them to the clerk, like:

“Please start a new file for John Doe - his telephone number is 555-1212”

“Update Jane’s address to be 123 Main St.”

When you send a multipart form request, it’s like handing a bunch of documents to the clerk.

Multipart Requests

HTTP Requests can also embed 1 or more files for uploading to a server. These requests are called multipart requests, because the request body has multiple parts:

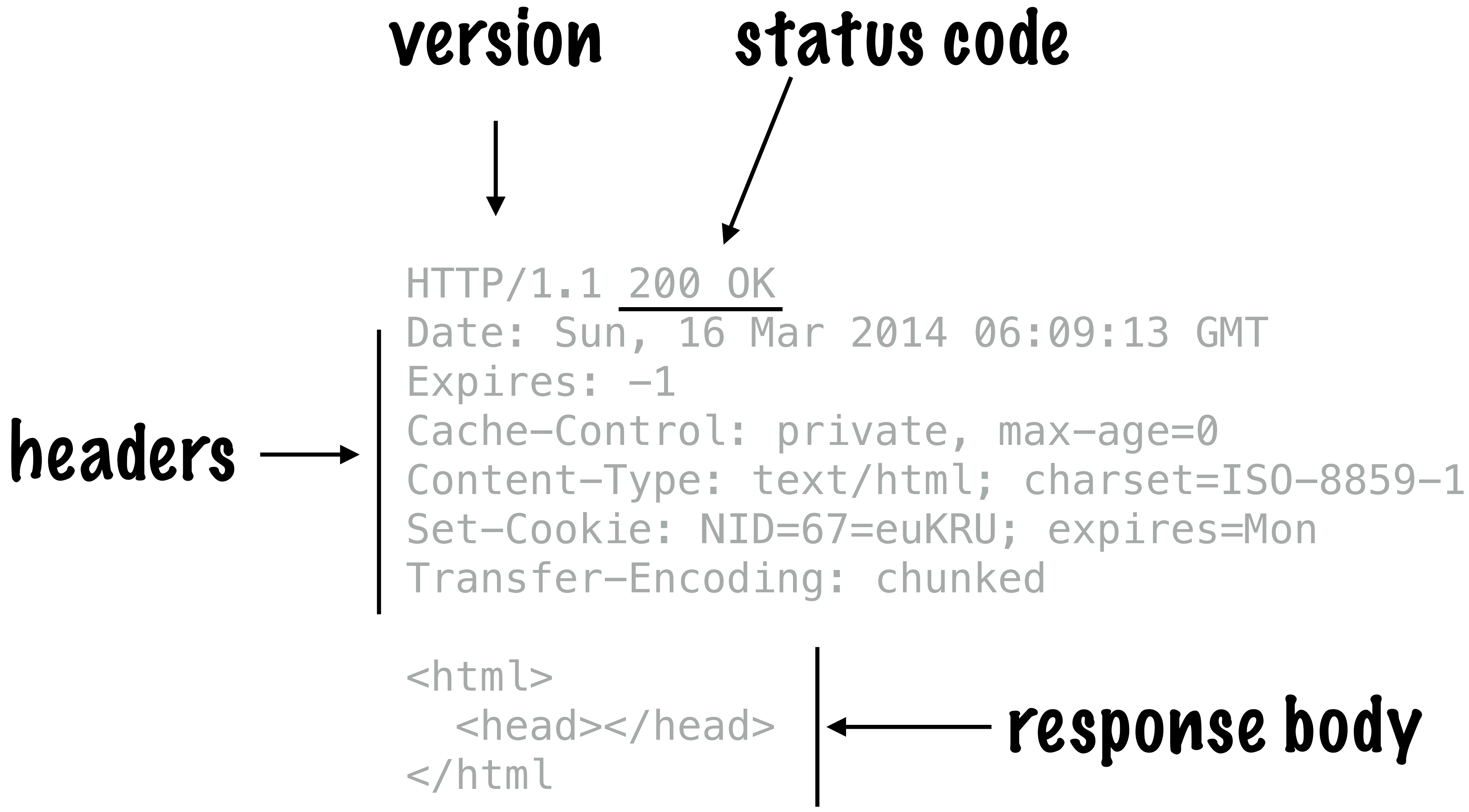
```
POST /accounts/ClientLogin HTTP/1.1
Host: www.google.com
Content-Type: multipart/form-data;
          boundary=gc0p4Jq0M2Yt08jU534c0p

--gc0p4Jq0M2Yt08jU534c0p
Content-Disposition: form-data; name="datafile1"; filename="r.gif"
Content-Type: image/gif

GIF87a…;
--gc0p4Jq0M2Yt08jU534c0p
```

HTTP Response

Response: Definition

An HTTP response is a snippet of text
that the server sends to the client,
which includes the HTTP version,
the status code, headers and
the response body.

Response: Example



```
HTTP/1.1 200 OK
Date: Sun, 16 Mar 2014 06:09:13 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
Set-Cookie: NID=67=euKRU; expires=Mon
Transfer-Encoding: chunked

<html>
  <head></head>
</html
```

Response Body

- Can be any text (HTML, XML, JSON etc…)
- Can include images, PDF and other binary file data
- Is interpreted based on the Content Type header

Status Codes

Each status code means something specific, and they are grouped into a ranges of 100:

2xx (success) - like `200 OK`

3xx (redirection) - like `301 Moved Permanently`

4xx (client error) - like `404 Not Found`

5xx (server error) - like `500 Internal Server Error`

Headers

Both Requests and Responses have headers. These are a dictionary of name / value pairs that add extra information about the request. Some common headers you'll see in almost every HTTP request / response are:

- Content Types
- Cookies
- User Agent
- Referer (sic)
- Character Sets
- etc…

Questions

- When you click “reload” in a browser, what’s happening under the hood (as far as requests / responses)

- Describe what happens when you click a “login” button on a website

References

- http://www.tutorialspoint.com/http/http_tutorial.pdf