Delivering web content

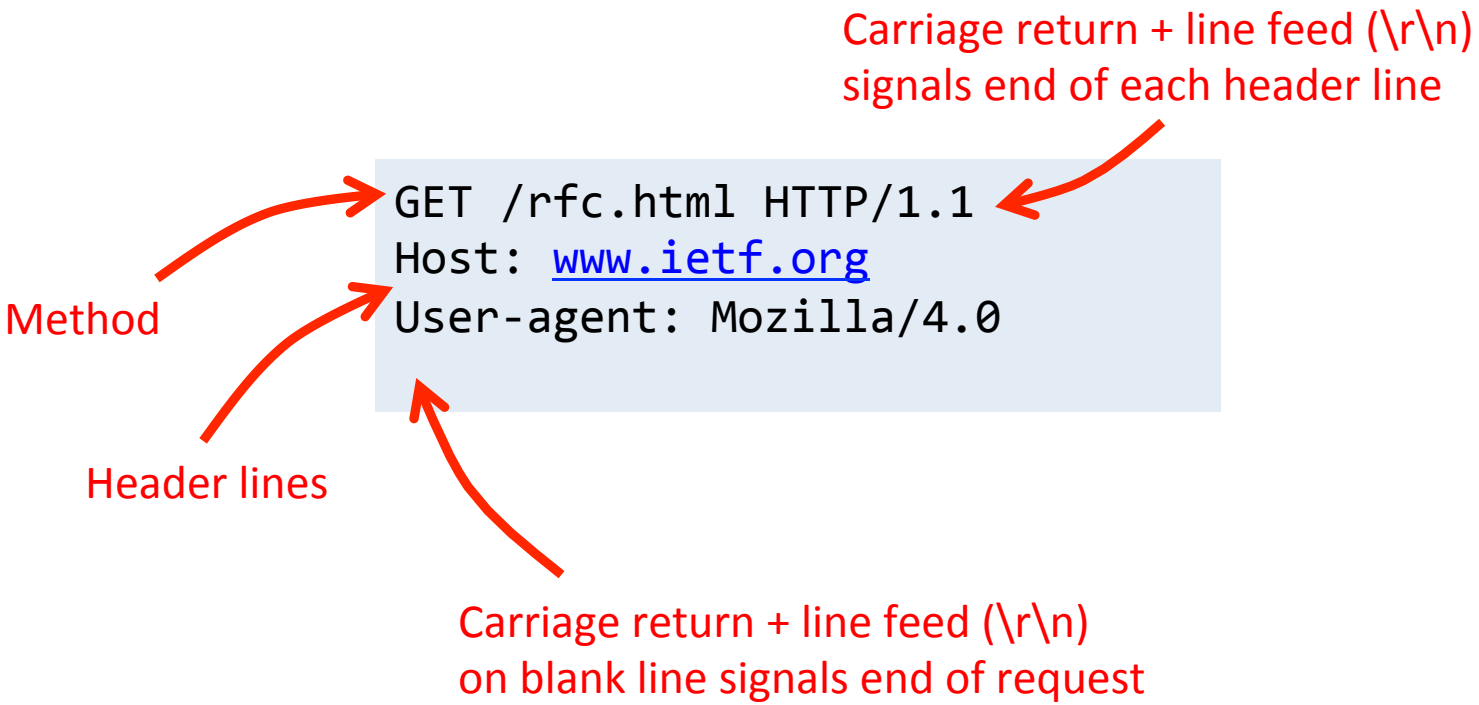
```
GET /rfc.html HTTP/1.1
Host: www.ietf.org
User-agent: Mozilla/4.0
```

Overview

- HTTP protocol review
  - Request and response format
  - GET versus POST
- Static and dynamic content
  - Client-side scripting
  - Server-side extensions
    - CGI
    - Server-side includes
    - Server-side scripting
    - Server modules
    - Servlets
    - …

HTTP protocol

- HyperText Transfer Protocol (HTTP)
  - Simple request-response protocol
  - Runs over TCP, port 80
  - ASCII format request and response headers



HTTP request

```
GET /rfc.html HTTP/1.1
Host: www.ietf.org
User-agent: Mozilla/4.0
```

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

```
POST /login.html HTTP/1.1
Host: www.store.com
User-agent: Mozilla/4.0
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

userid=joe&password=guessme
```

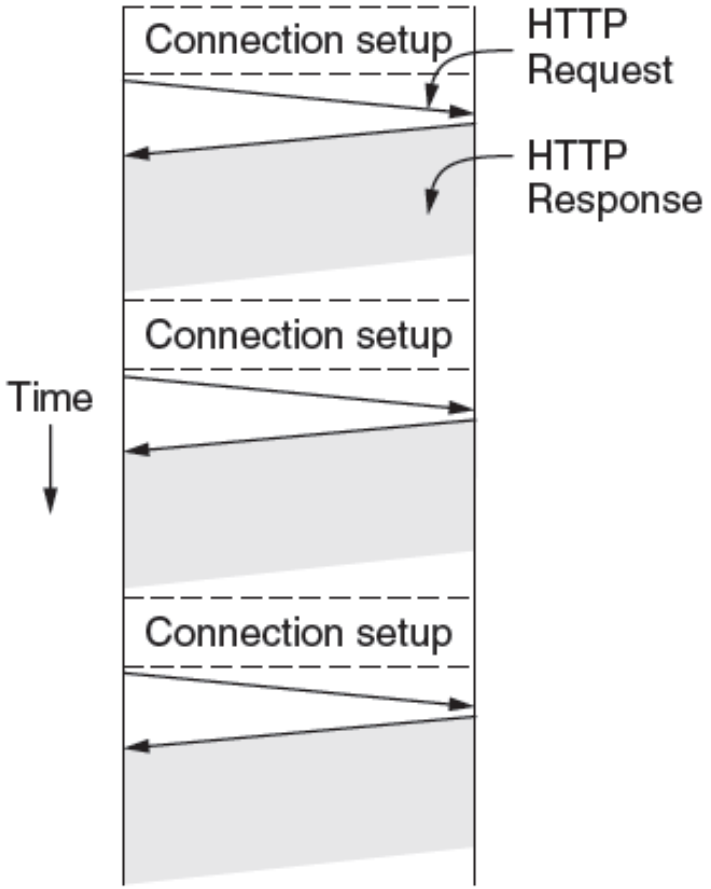
HTTP response

- Response from server
  - Status line: protocol version, status code & phrase
  - Response headers: extra info
  - Body: optional data

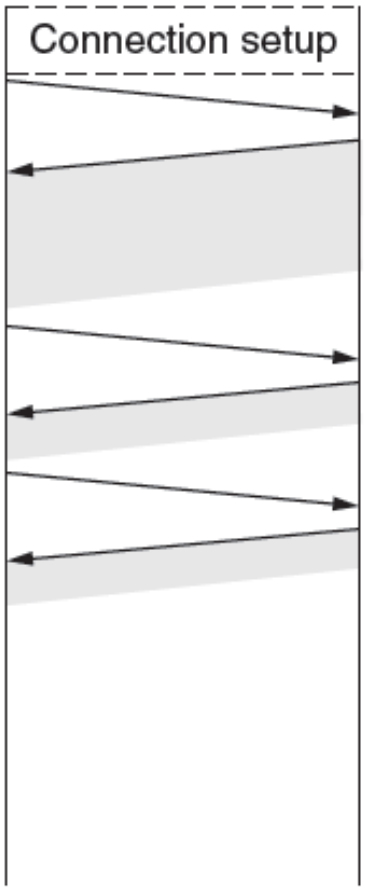
```
HTTP/1.1 200 OK
Date: Thu, 17 Nov 2011 15:54:10 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Wed, 14 Sep 2011 17:04:27 GMT
Content-Length: 285

<html> …
```

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

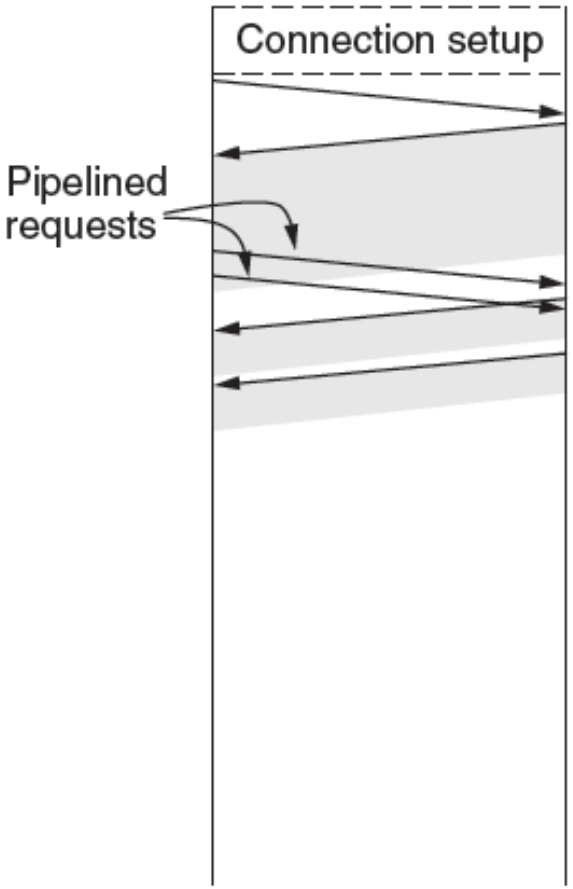
TCP details



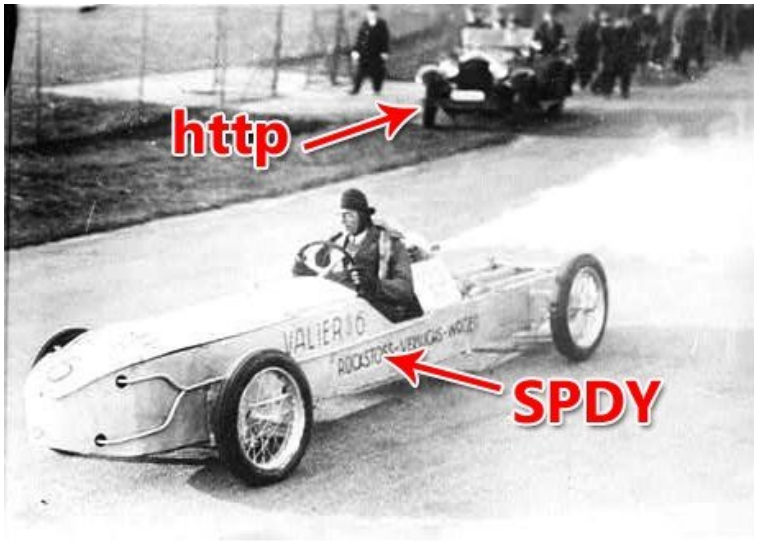
Multiple connections and sequential requests.



Persistent connection and sequential requests.



Persistent connection and pipelined requests.



rocket car courtesy Wikipedia

http://downloadsquad.switched.com/2009/11/12/google-speed-up-web-chrome-os-with-spdy/

"64% reductions in page load times in SPDY"

-Google

https://www.youtube.com/watch?v=WkLBrHW4NhQ

https://www.youtube.com/watch?v=E9FxNzv1Tr8

Hypertext Transfer Protocol version 2
draft-ietf-httpbis-http2-latest

Abstract

This specification describes an optimized expression of the semantics of the Hypertext Transfer Protocol (HTTP). HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing header field compression and allowing multiple concurrent messages on the same connection. It also introduces unsolicited push of representations from servers to clients.

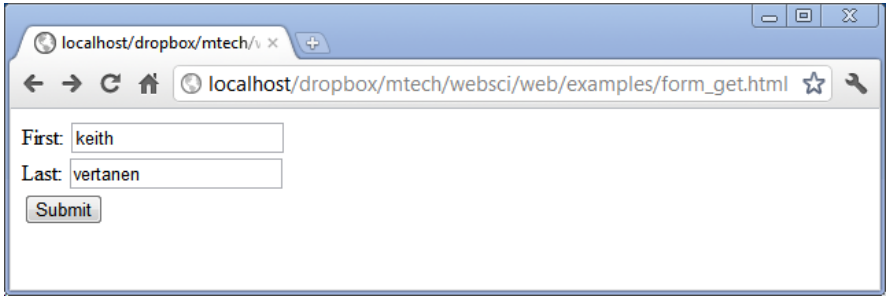
This specification is an alternative to, but does not obsolete, the HTTP/1.1 message syntax. HTTP's existing semantics remain unchanged.

GET versus POST

- Two ways to send input to web server
  - GET and POST

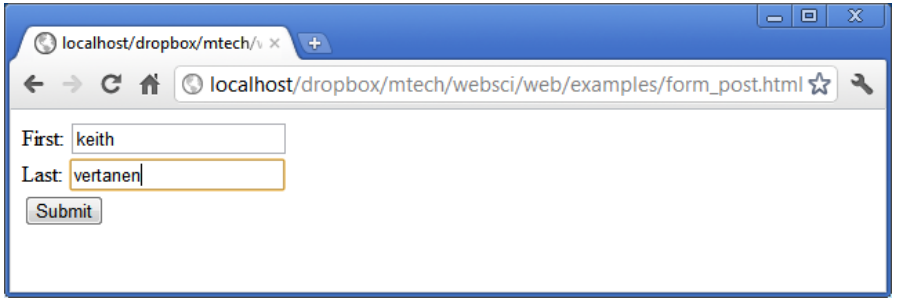
GET

```
<html>
<body>
<form action="submit.html" method="GET">
First: <input type="text" name="first" /><br />
Last: <input type="text" name="last" /><br />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```



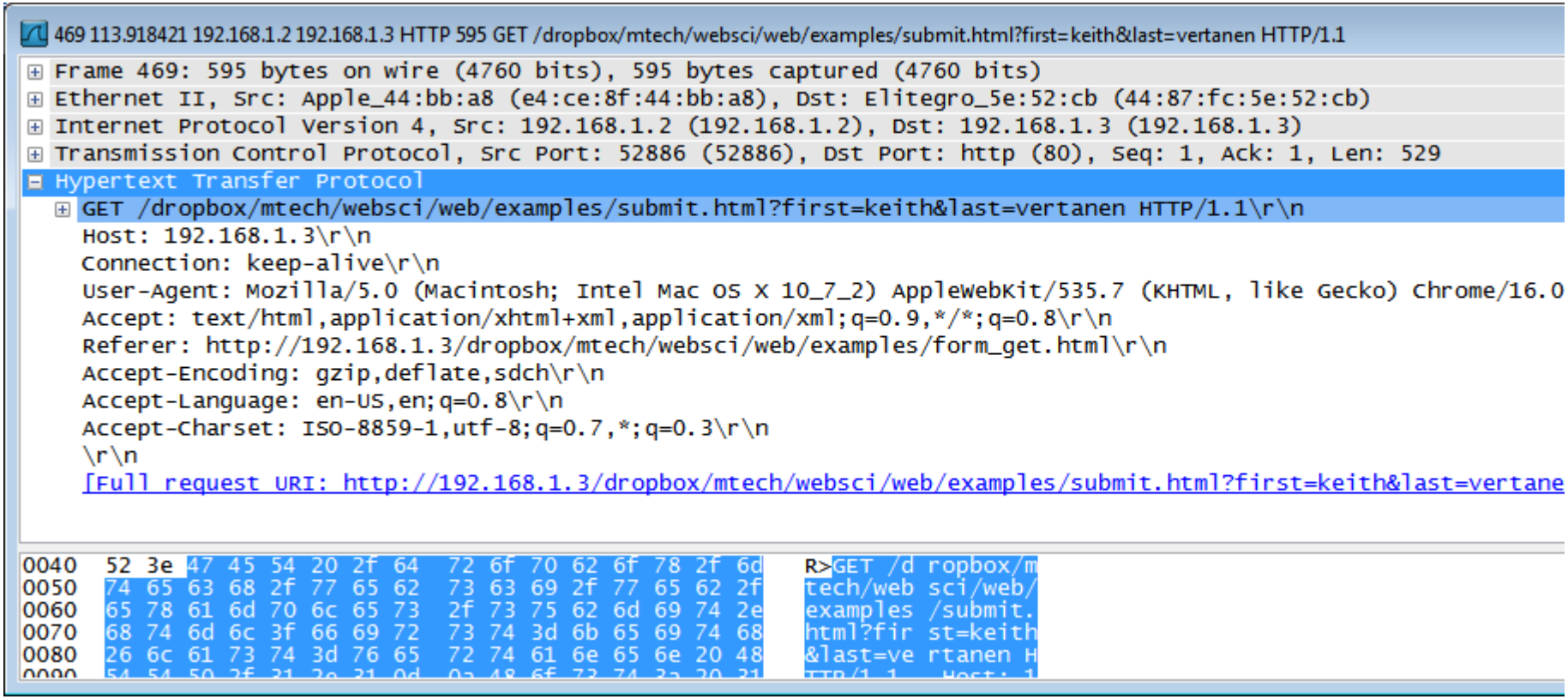
POST

```
<html>
<body>
<form action="submit.html" method="POST">
First: <input type="text" name="first" /><br />
Last: <input type="text" name="last" /><br />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```



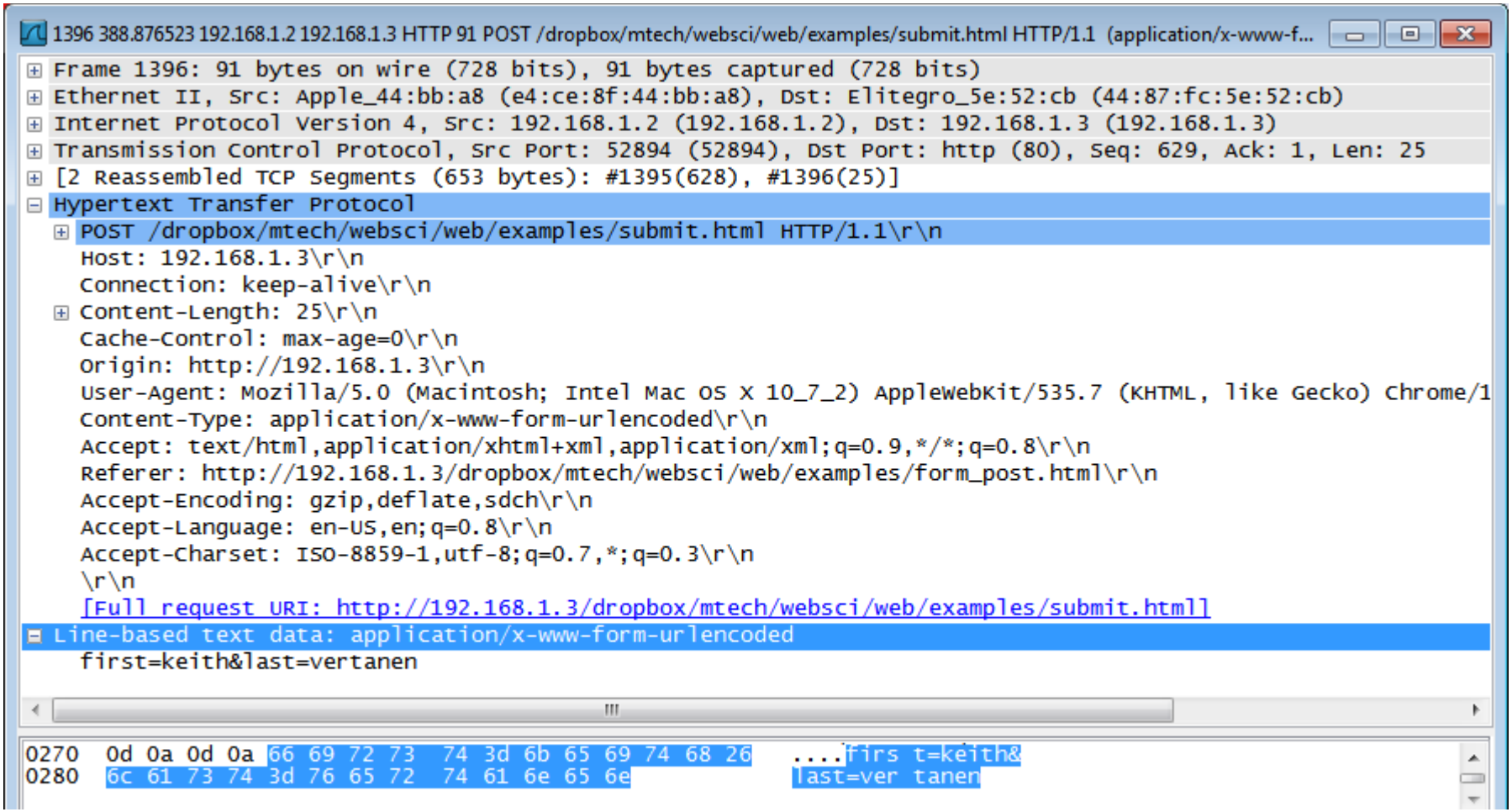
GET

```
<html>
<body>
<form action="submit.html" method="GET">
First: <input type="text" name="first" /><br />
Last: <input type="text" name="last" /><br />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```



POST

```
<html>
<body>
<form action="submit.html" method="POST">
First: <input type="text" name="first" /><br />
Last: <input type="text" name="last" /><br />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```



Let's build a web server

- Simple Java web server
  - Only handle GET requests
    - Return entire page (no conditional requests)
    - 404 if page doesn't exist
  - Only look at first line of HTTP message
    - Ignore all headers
  - Only serve text pages
    - No images
    - No dynamic content
  - Multithreaded

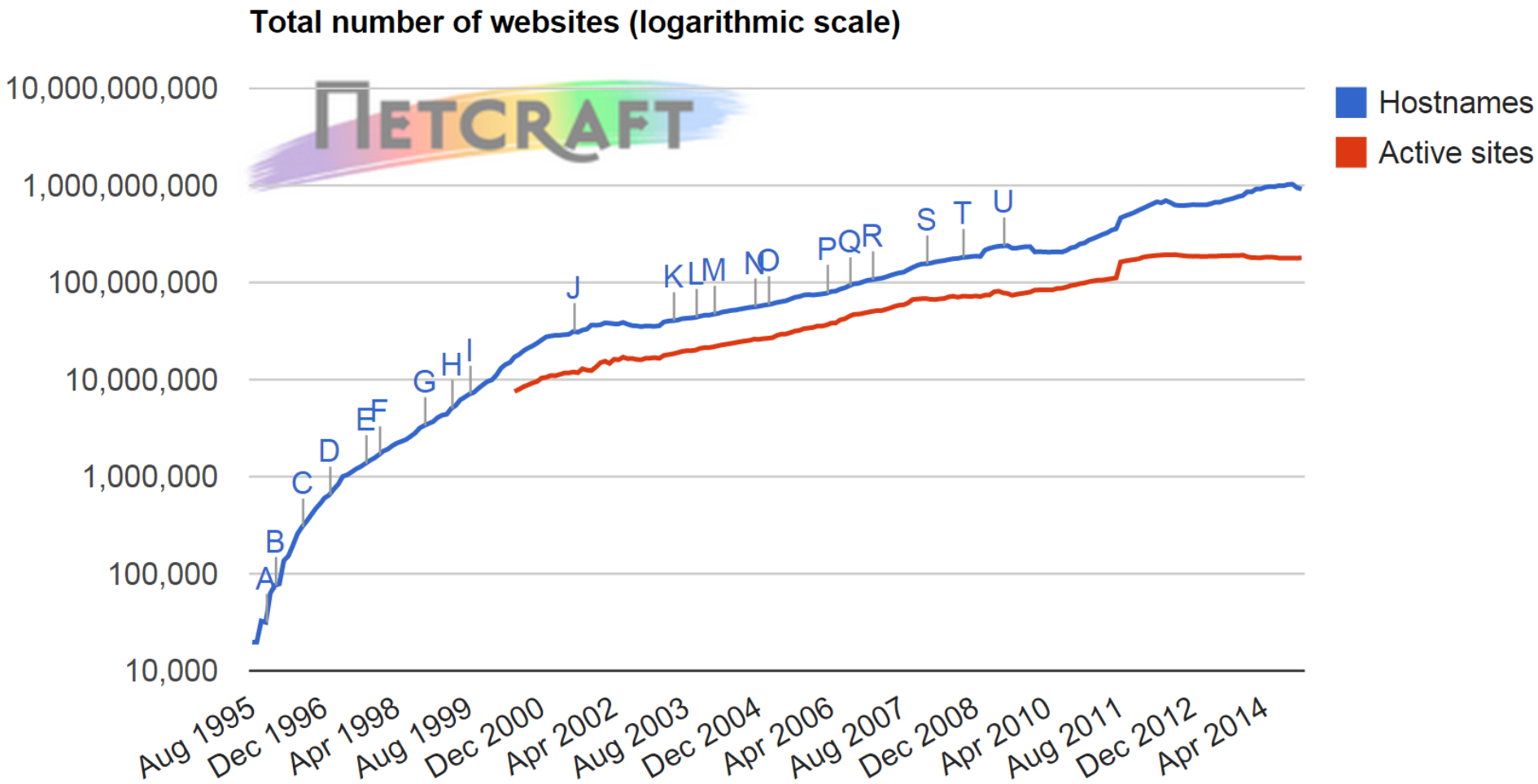
```
GET /index.html HTTP/1.0
Host: www.blah.com
User-agent: Mozilla/4.0
```

```
HTTP/1.0 200 OK

<html> …
```

Server	Developed by	Software license	Last stable version	Latest release date
AOLserver	NaviSoft	Mozilla	4.5.2	2012-09-19
Apache HTTP Server	Apache Software Foundation	Apache	2.4.10	2014-07-21
Apache Tomcat	Apache Software Foundation	Apache	8.0.15	2014-11-07
Boa	Paul Phillips	GPL	0.94.13	2002-07-30
Caudium	The Caudium Group	GPL	1.4.18	2012-02-24
Cherokee HTTP Server	Álvaro López Ortega	GPL	1.2.103	2013-04-21
Hiawatha HTTP Server	Hugo Leisink	GPLv2	9.9	2014-12-07
HFS	Rejetto	GPL	2.2f	2009-02-17
IBM HTTP Server	IBM	Non-free proprietary	8.5.5	2013-06-14
Internet Information Services	Microsoft	Non-free proprietary	8.5	2013-09-09
Jetty	Eclipse Foundation	Apache	9.1.4	2014-04-01
Jexus	Bing Liu	Non-free proprietary	5.5.2	2014-04-27
lighttpd	Jan Kneschke (Incremental)	BSD variant	1.4.35	2014-03-12
LiteSpeed Web Server	LiteSpeed Technologies	Non-free proprietary	4.2.3	2013-05-22
Mongoose	Cesanta Software	GPLv2 / commercial	5.5	2014-10-28
Monkey HTTP Server	Monkey Software	Apache	1.5.1	2014-06-10
NaviServer	Various	Mozilla 1.1	4.99.6	2014-06-29
NCSA HTTPd	Robert McCool	Non-free proprietary	1.5.2a	1996
Nginx	NGINX, Inc.	BSD variant	1.7.8	2014-12-02
OpenBSD httpd	Reyk Floeter, OpenBSD	ISC		2014-10-09
OpenLink Virtuoso	OpenLink Software	GPL and commercial versions	7.1.0	2014-02-17
OpenLiteSpeed	LiteSpeed Technologies	GPLv3 / commercial	1.3.2	2014-05-22
Oracle HTTP Server	Oracle Corporation	Non-free proprietary	12.1.2	2013
Oracle iPlanet Web Server	Oracle Corporation	BSD	7.0.19	2014-01-14
Oracle WebLogic Server	Oracle Corporation (formerly BEA Systems)	Non-free proprietary	12cR3 (12.1.3)	2014-06-26
Resin Open Source	Caucho Technology	GPLv3 / commercial	4.0.39	2014-04-07
Resin Professional	Caucho Technology	Non-free proprietary	4.0.39	2014-04-07
thttpd	Jef Poskanzer for ACME Laboratories	BSD variant	2.25b	2003-12-29
TUX web server	Ingo Molnár	GPL	3.2.6.18	2006-09-20
Wakanda Server	4D	AGPLv3 / Commercial	8.159169	2014-04-10
WEBrick	Ruby Community	Ruby	1.9.3 p286 (Ruby)	2012-10-12
Xitami	iMatix Corporation	BSD	5.0a0	2009-02-19
Yaws	Claes Wikström	BSD variant	1.98	2013-11-04
Zeus Web Server	Zeus Technology	Non-free proprietary	4.3r5	2010-01-13
Zope	Zope Corporation	Zope	2.13.21	2013-07-16
Server	**Creator**	**Software license**	**Last stable version**	**Release date**

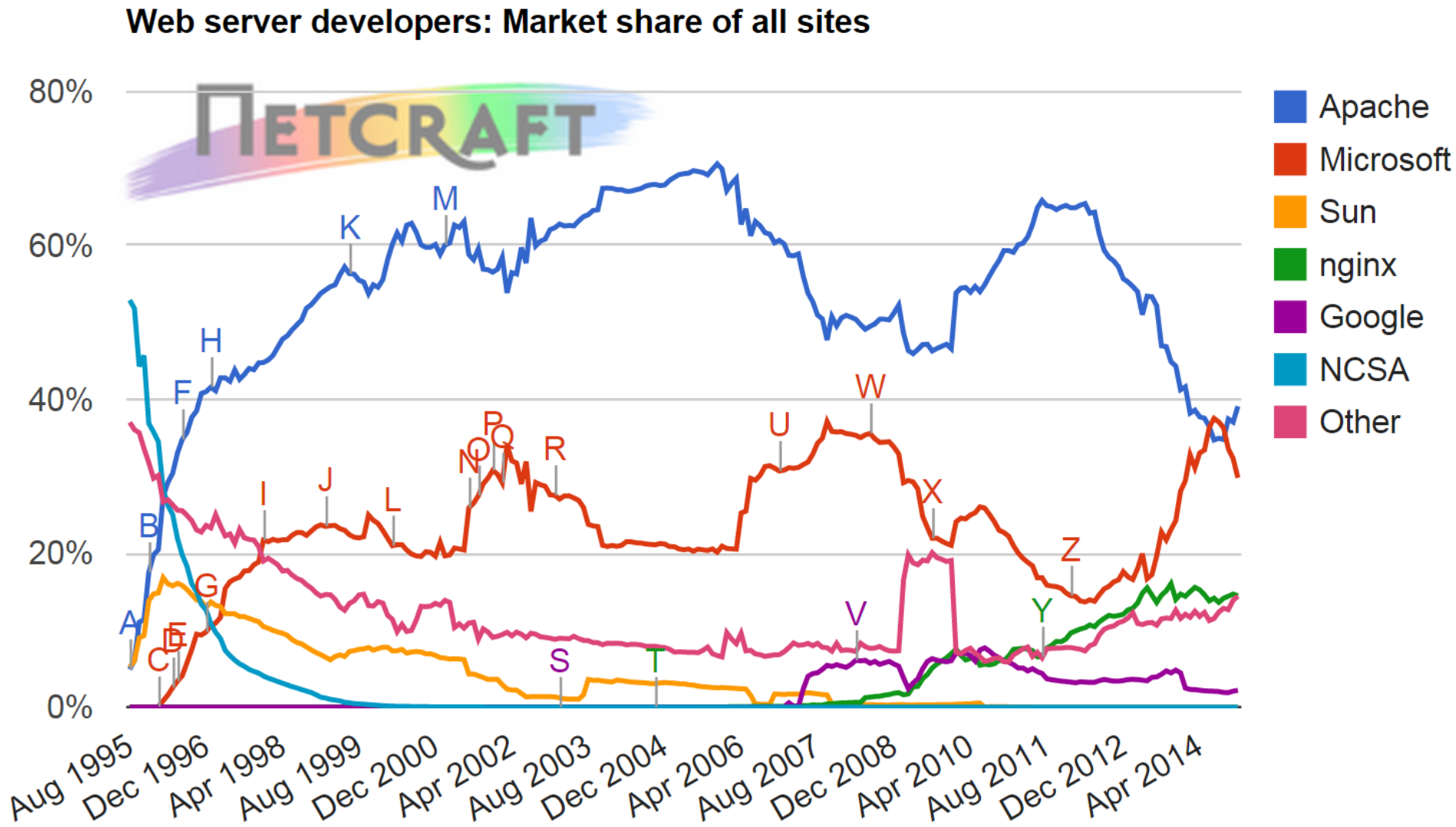
http://en.wikipedia.org/wiki/Comparison_of_web_server_software

Total number of websites

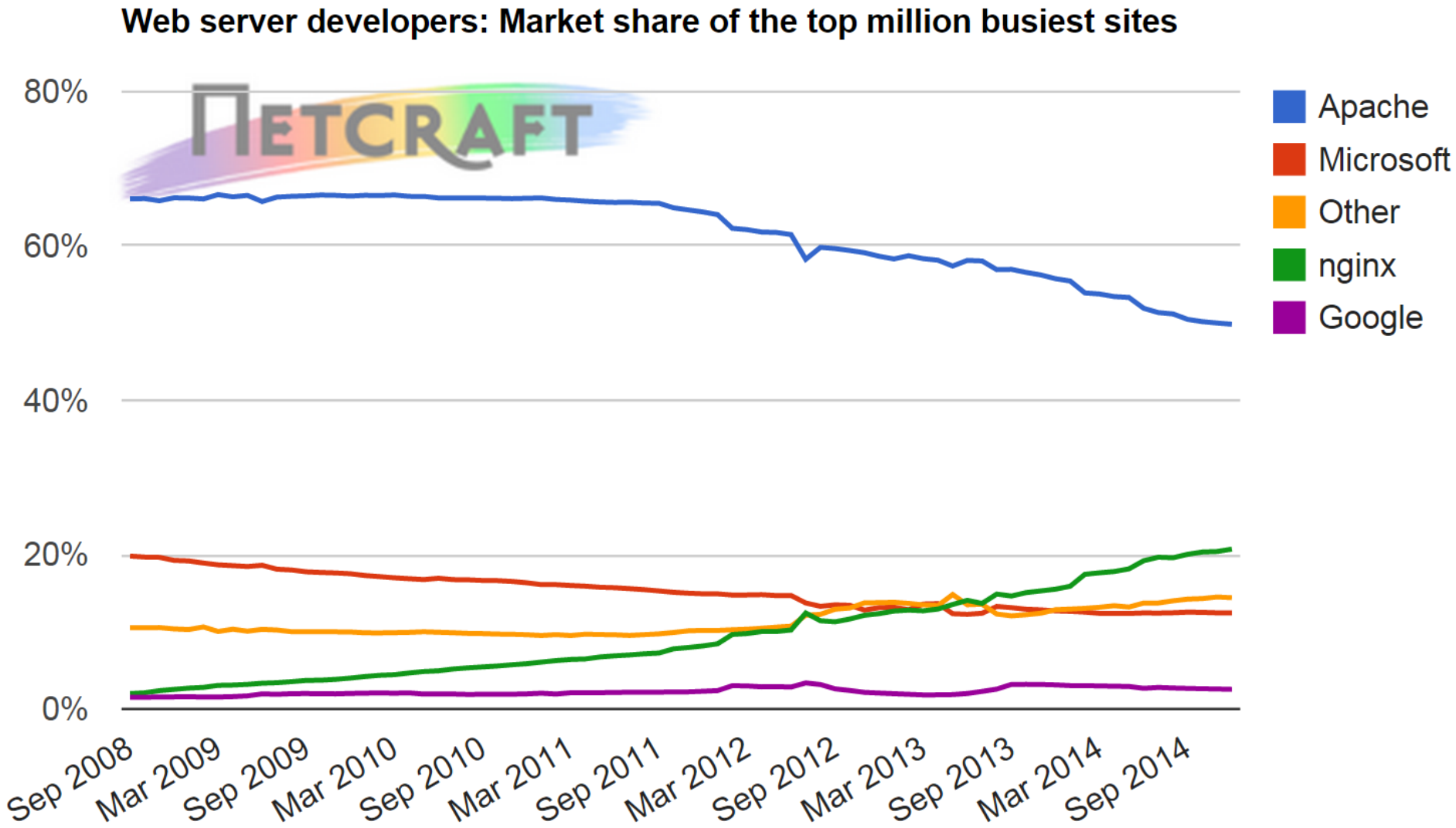


http://news.netcraft.com/archives/2014/12/18/december-2014-web-server-survey.html

Web servers in use



Web servers in use: busiest sites



nginx???

- nginx
  - "engine-x"
    - Originally designed for Russian Rambler sites
    - 500M request/day
    - Free, open-source
    - Facebook, Zappos, Hulu, Dropbox, Wordpress…
  - Design:
    - High performance, high concurrency, low memory
    - Event-driven instead of using threads
    - C10K problem: 10,000 simultaneous connections

"Apache is like Microsoft Word, it has a million options but you only need six. Nginx does those six things, and it does five of them 50 times faster than Apache."

-Chris Lea

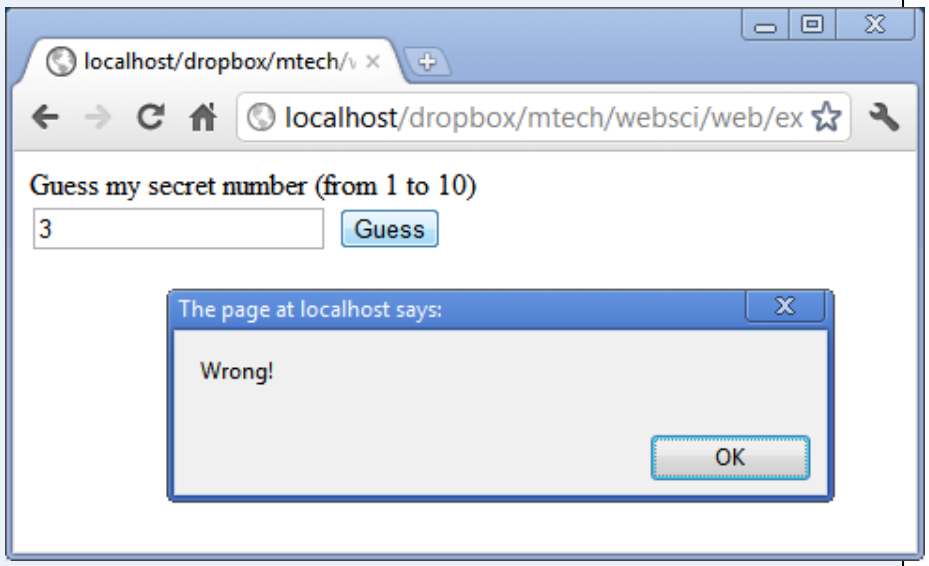
Static vs. dynamic

- Static content
  - Images and pages don't change
    - Web server is like a file server
  - Fast to deliver, easy to cache
- Dynamic content
  - Same URL may result in delivery of different HTML
    - e.g. preference on # of products to display
    - e.g. user adds items to a shopping cart
  - Need something besides HTTP + HTML
    - HTTP is stateless
    - HTTP not programmable (no conditionals, loops, etc.)

Scripting

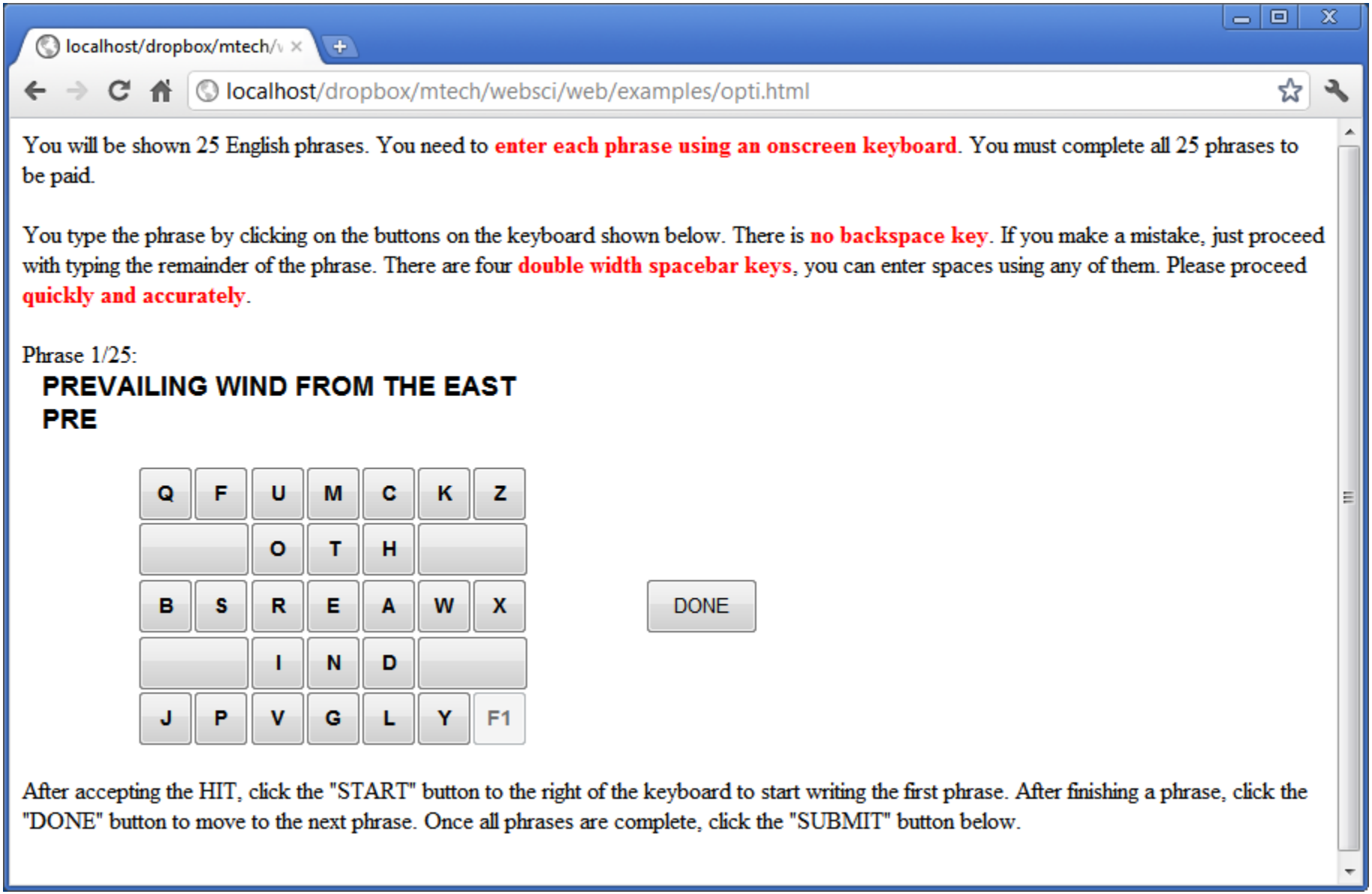
- Client-side scripting
  - Runs in the client's browser
    - Fast, no roundtrip to the server
  - Users can see if they "view source"
  - De facto client-side language: JavaScript
    - Heavily influenced by Java, but not the same
    - Weakly typed
  - The "dynamic" in DHTML
    - e.g. validate form fields, mouse over effects
  - Exact behavior may depend on browser

```
<html>
<head>
<script language="JavaScript">
var secret = Math.floor(Math.random() * 10) + 1;

function check()
{
	if (document.forms.game.guess.value == secret)
	{
		alert('Well done!');
	}
	else
	{
		alert('Wrong!');
		document.forms.game.guess.value = "";
	}
}
</script>
</head>
<body>

Guess my secret number (from 1 to 10)<br />
<form onSubmit="check()" name="game">
<input type="text" name="guess" VALUE="">
<input type="button" value="Guess" onClick='check()'>
</form>
</body>
</html>
```



JavaScript

- Sophisticated apps can be built on single page
  - Saves on network roundtrips
    - Very responsive for the user
  - Reduces server load from many small requests
    - One big request
  - Can be difficult to debug
    - JavaScript browser console = big help
  - JavaScript tied to browser / client
    - Stat lost if browser closed, crashes, page reloaded
    - Speed may vary with browser and client hardware
      - e.g. mobile device


localhost/dropbox/mtech/
localhost/dropbox/mtech/websci/web/examples/opti.html
You will be shown 25 English phrases. You need to enter each phrase using an onscreen keyboard. You must complete all 25 phrases to be paid.
You type the phrase by clicking on the buttons on the keyboard shown below. There is no backspace key. If you make a mistake, just proceed with typing the remainder of the phrase. There are four double width spacebar keys, you can enter spaces using any of them. Please proceed quickly and accurately.
Phrase 1/25:
PREVAILING WIND FROM THE EAST
PRE
Q F U M C K Z
O T H
B S R E A W X
I N D
J P V G L Y F1
DONE
After accepting the HIT, click the "START" button to the right of the keyboard to start writing the first phrase. After finishing a phrase, click the "DONE" button to move to the next phrase. Once all phrases are complete, click the "SUBMIT" button below.

Web server extensions

- Web server extensions
  - Functionality on top of serving static HTML pages
  - Implemented by the web server
    - e.g. PHP, ASP, ColdFusion, JSP, SSI, CGI, FastCGI, SCGI, ISAPI, Apache modules (mod_perl, mod_python)
- Advantages:
  - Store data and runs on server
    - e.g. shared database of products
  - Code details hidden from client
    - Browser sees resulting HTML, not how it was generated
  - Improved code maintainability
    - Put repeated HTML (e.g. header/footer) into a single file

CGI

- CGI (Common Gateway Interface)
  - In use since 1993
  - URL request in a special location/file extension
    - e.g. `http://blah.com/cgi.bin/lookup`
  - Web server passes request to script/program
    - Sets a laundry list of environment variables
    - Creates new process and runs program
    - Program's output sent to client
  - Notes:
    - Needs read+execute permissions for web server's user
    - Probably shouldn't be world writeable

CGI in Apache

```
<IfModule mod_alias.c>
        <IfModule mod_cgi.c>
                Define ENABLE_USR_LIB_CGI_BIN
        </IfModule>

        <IfModule mod_cgid.c>
                Define ENABLE_USR_LIB_CGI_BIN
        </IfModule>

        <IfDefine ENABLE_USR_LIB_CGI_BIN>
                ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
                <Directory "/usr/lib/cgi-bin">
                        AllowOverride None
                        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                        Require all granted
                </Directory>
        </IfDefine>
</IfModule>
```

server-cgi-bin.conf

```
#!/usr/bin/perl

print "Content-type: text/plain;
charset=iso-8859-1\n\n";
foreach $var (sort(keys(%ENV))) {
    $val = $ENV{$var};
    $val =~ s|\n|\\n|g;
    $val =~ s|"|\\"|g;
    print "${var}=\"${val}\"\n";
}
```

cgi-bin/printenv.pl

```
DOCUMENT_ROOT="/usr/local/apache2/htdocs"
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
HTTP_ACCEPT_CHARSET="ISO-8859-1,utf-8;q=0.7,*;q=0.3"
HTTP_ACCEPT_ENCODING="gzip,deflate,sdch"
HTTP_ACCEPT_LANGUAGE="en-US,en;q=0.8"
HTTP_CACHE_CONTROL="max-age=0"
HTTP_CONNECTION="keep-alive"
HTTP_HOST="localhost"
HTTP_USER_AGENT="Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2)
AppleWebKit/535.7 (KHTML, like Gecko) Chrome/16.0.912.75 Safari/535.7"
PATH=".:/Users/kvertanen/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin"
QUERY_STRING=""
REMOTE_ADDR="::1"
REMOTE_PORT="53160"
REQUEST_METHOD="GET"
REQUEST_URI="/cgi-bin/printenv"
SCRIPT_FILENAME="/usr/local/apache2/cgi-bin/printenv"
SCRIPT_NAME="/cgi-bin/printenv"
SERVER_ADDR="::1"
SERVER_ADMIN="you@example.com"
SERVER_NAME="localhost"
SERVER_PORT="80"
SERVER_PROTOCOL="HTTP/1.1"
SERVER_SIGNATURE=""
SERVER_SOFTWARE="Apache/2.2.21 (Unix) PHP/5.3.9"
```

http://localhost/cgi-bin/printenv.pl

```
DOCUMENT_ROOT="/usr/local/apache2/htdocs"
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
HTTP_ACCEPT_CHARSET="ISO-8859-1,utf-8;q=0.7,*;q=0.3"
HTTP_ACCEPT_ENCODING="gzip,deflate,sdch"
HTTP_ACCEPT_LANGUAGE="en-US,en;q=0.8"
HTTP_CACHE_CONTROL="max-age=0"
HTTP_CONNECTION="keep-alive"
HTTP_HOST="localhost"
HTTP_USER_AGENT="Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2)
AppleWebKit/535.7 (KHTML, like Gecko) Chrome/16.0.912.75 Safari/535.7"
PATH=".:/Users/kvertanen/bin:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin"
QUERY_STRING="foo=bar&hello=world"
REMOTE_ADDR="::1"
REMOTE_PORT="53160"
REQUEST_METHOD="GET"
REQUEST_URI="/cgi-bin/printenv"
SCRIPT_FILENAME="/usr/local/apache2/cgi-bin/printenv"
SCRIPT_NAME="/cgi-bin/printenv"
SERVER_ADDR="::1"
SERVER_ADMIN="you@example.com"
SERVER_NAME="localhost"
SERVER_PORT="80"
SERVER_PROTOCOL="HTTP/1.1"
SERVER_SIGNATURE=""
SERVER_SOFTWARE="Apache/2.2.21 (Unix) PHP/5.3.9"
```

http://localhost/cgi-bin/printenv.pl?foo=bar&hello=world

CGI in C

- CGI in C
  - Use `getenv()` to get environment parameters
  - If target of GET form: use QUERY_STRING
  - If target of POST form: read from standard input

```
#include <stdio.h>
#include <stdlib.h>  // don't forget this!

int main(void)
{
   printf("Content-Type: text/plain;charset=us-ascii\n\n");
   printf("Hello world!\n");
   if (getenv("QUERY_STRING") != NULL)
      printf("Query = %s\n", getenv("QUERY_STRING"));
   return 0;
}
```

Benchmarking web server

- Apache benchmark (ab)

```
/usr/local/apache2/bin/ab [options] [URL]
    -n  number of requests
    -c  concurrent requests
    [-k  make multiple request on same HTTP connection]
```

```
ab -n 100 -c 1 'http://127.0.0.1/index.html'
```

```
ab -n 100 -c 1 'http://127.0.0.1/cgi-bin/lookup?file=test100&name=hello&val=world'
```

```
ab -n 10000 –c 10 'http://127.0.0.1/cgi-bin/lookup?file=test100&name=hello'
```

```
keith@pizza:~$ ab -n 1000 -c 40 -k 'http://123.123.123.123/index.html'
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 104.236.117.243 (be patient)
Completed 100 requests
...
Finished 1000 requests

Server Software:        Apache/2.4.7
Server Hostname:        104.236.117.243
Server Port:            80

Document Path:          /index.html
Document Length:        11510 bytes

Concurrency Level:      40
Time taken for tests:   5.751 seconds
Complete requests:      1000
Failed requests:        0
Keep-Alive requests:    1000
Total transferred:      11819040 bytes
HTML transferred:       11510000 bytes
Requests per second:    173.89 [#/sec] (mean)
Time per request:       230.024 [ms] (mean)
Time per request:       5.751 [ms] (mean, across all concurrent requests)
Transfer rate:          2007.10 [Kbytes/sec] received

...
```

Summary

- HTTP protocol
  - Stateless request/response protocol
- Web servers
  - Built a simple Java web server
  - Servers in the wild: Apache/IIS/nginx dominate
- Web server extensions
  - Many choices with different tradeoffs, one choice: CGI
- Benchmarking using Apache benchmark: ab