Delivering web content

GET /rfc.html HTTP/1.1

GET /rfc.html HTTP/1.1

GET /rfc.html HTTP/1.1

GET /rfc.html HTTP/1.1

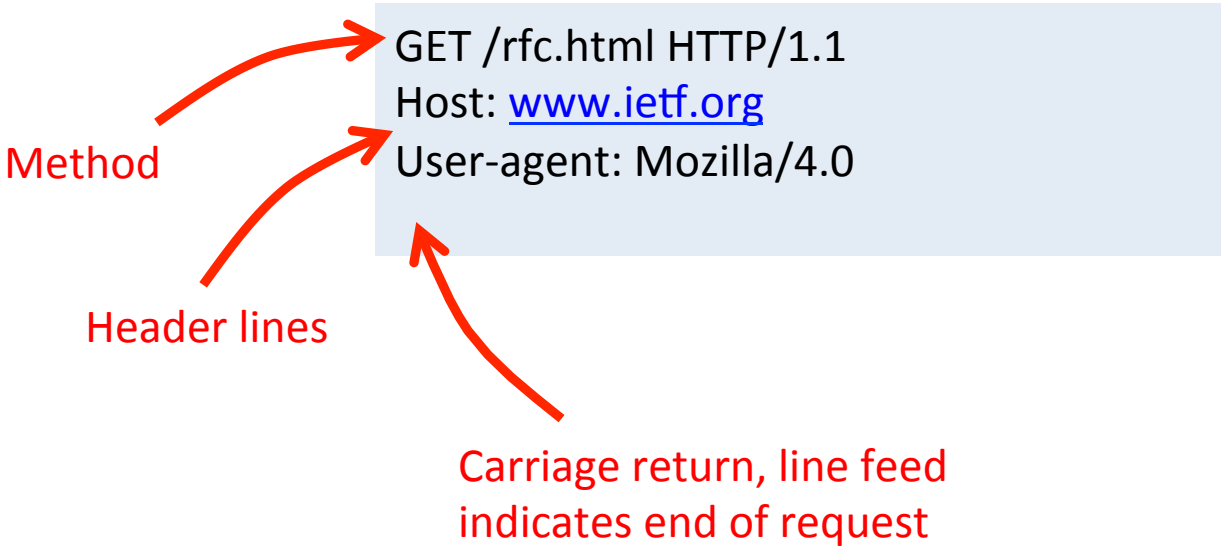
GET /rfc.html HTTP/1.1
Host: www.ietf.org
User-agent: Mozilla/4.0

Overview

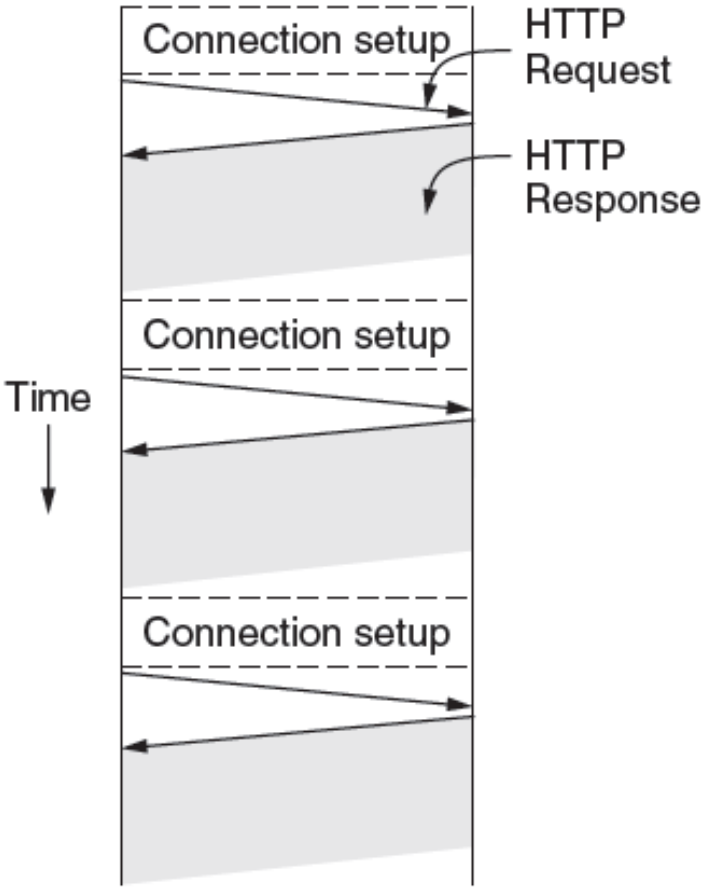
- HTTP protocol review
  - Request and response format
  - GET versus POST
- Static and dynamic content
  - Client-side scripting
  - Server-side extensions
    - CGI
    - Server-side includes
    - Server-side scripting
    - Server modules
    - Servlets
    - …

HTTP protocol

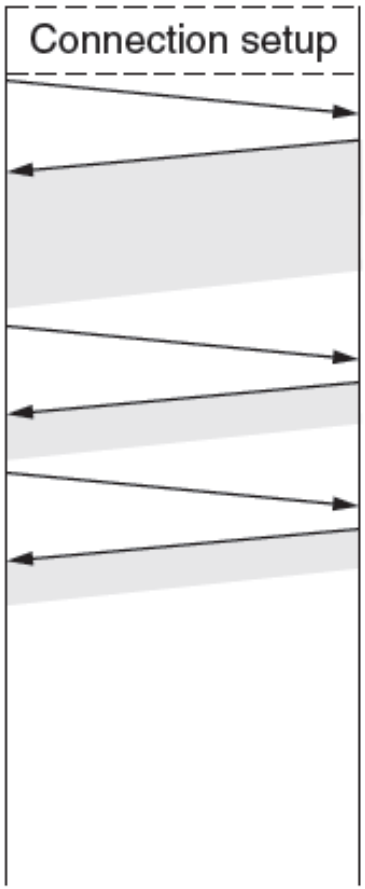
- HyperText Transfer Protocol (HTTP)
  - Simple request-response protocol
  - Runs over TCP, port 80
  - ASCII format request and response headers



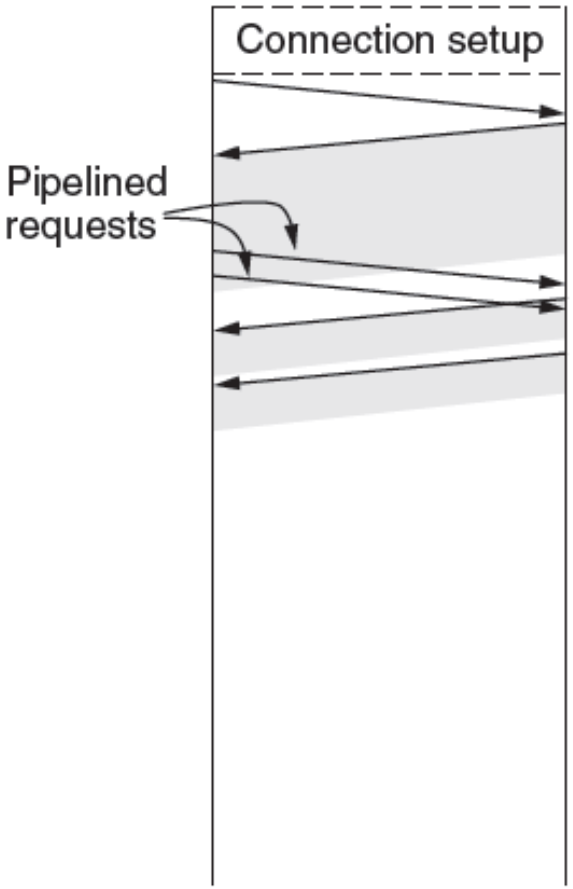
TCP details



Multiple connections and sequential requests.



Persistent connection and sequential requests.



Persistent connection and pipelined requests.

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, status 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> …
```

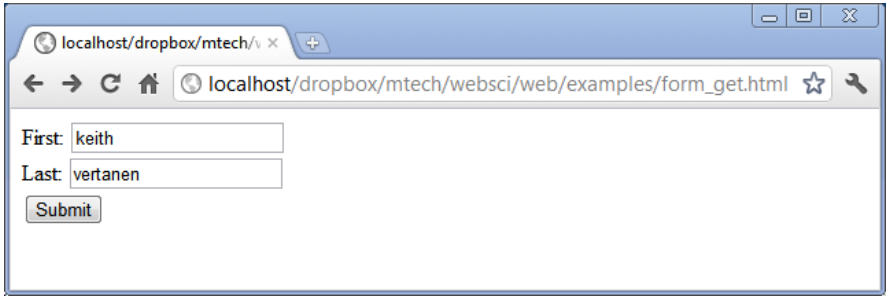
Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal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

GET versus POST

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

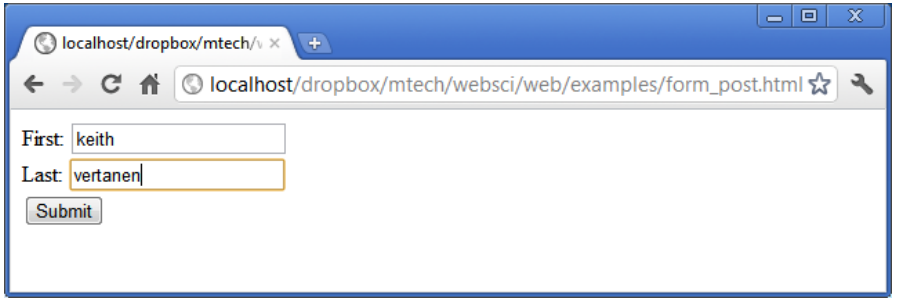
GET

```
<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>
```



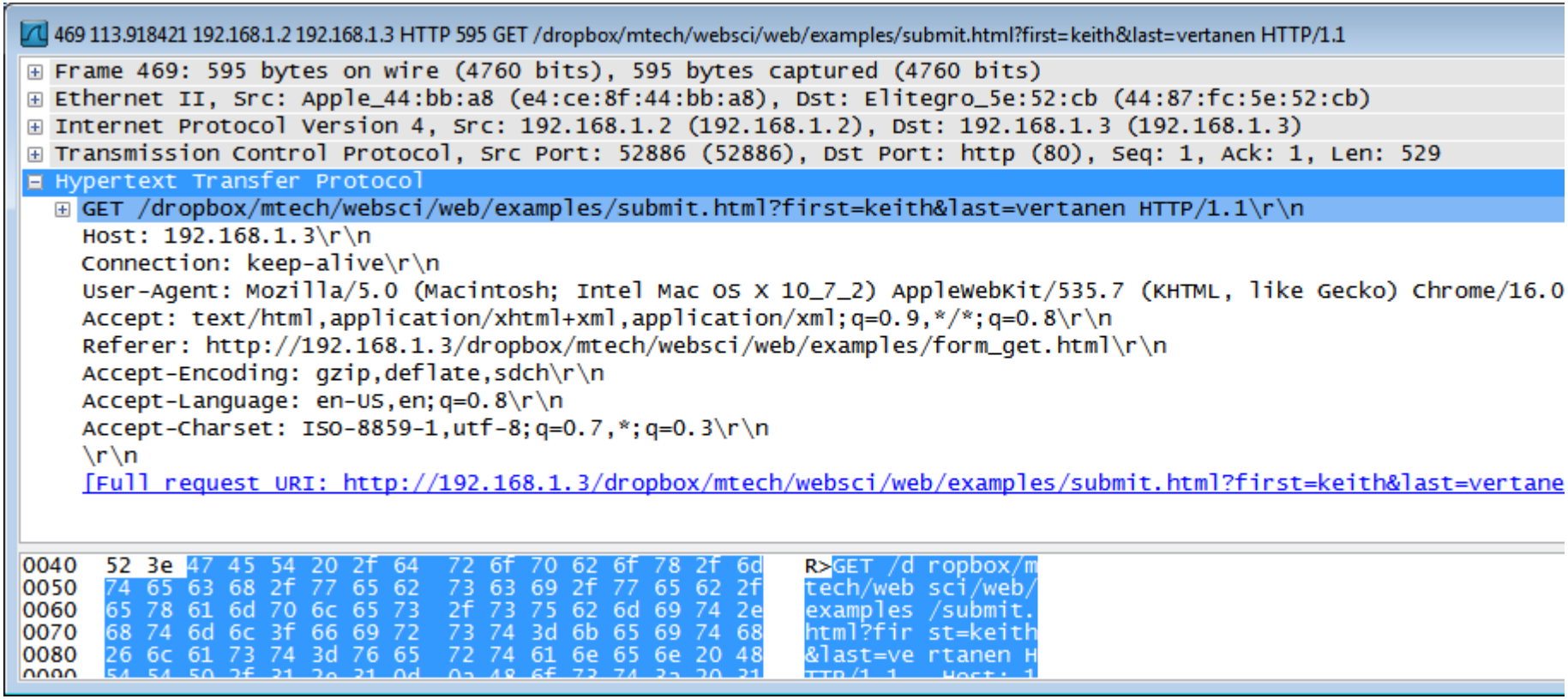
POST

```
<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>
```



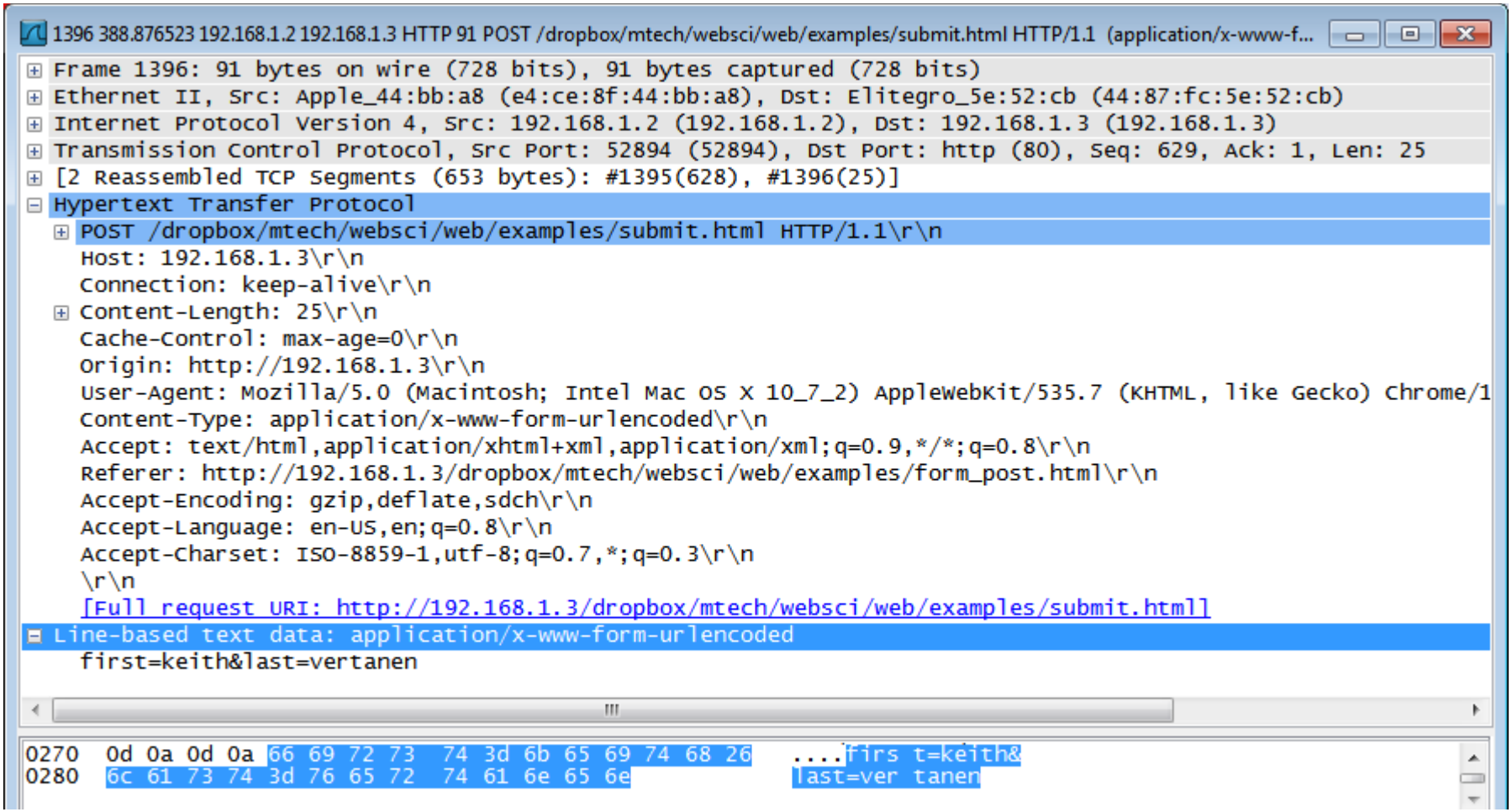
GET

```
<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

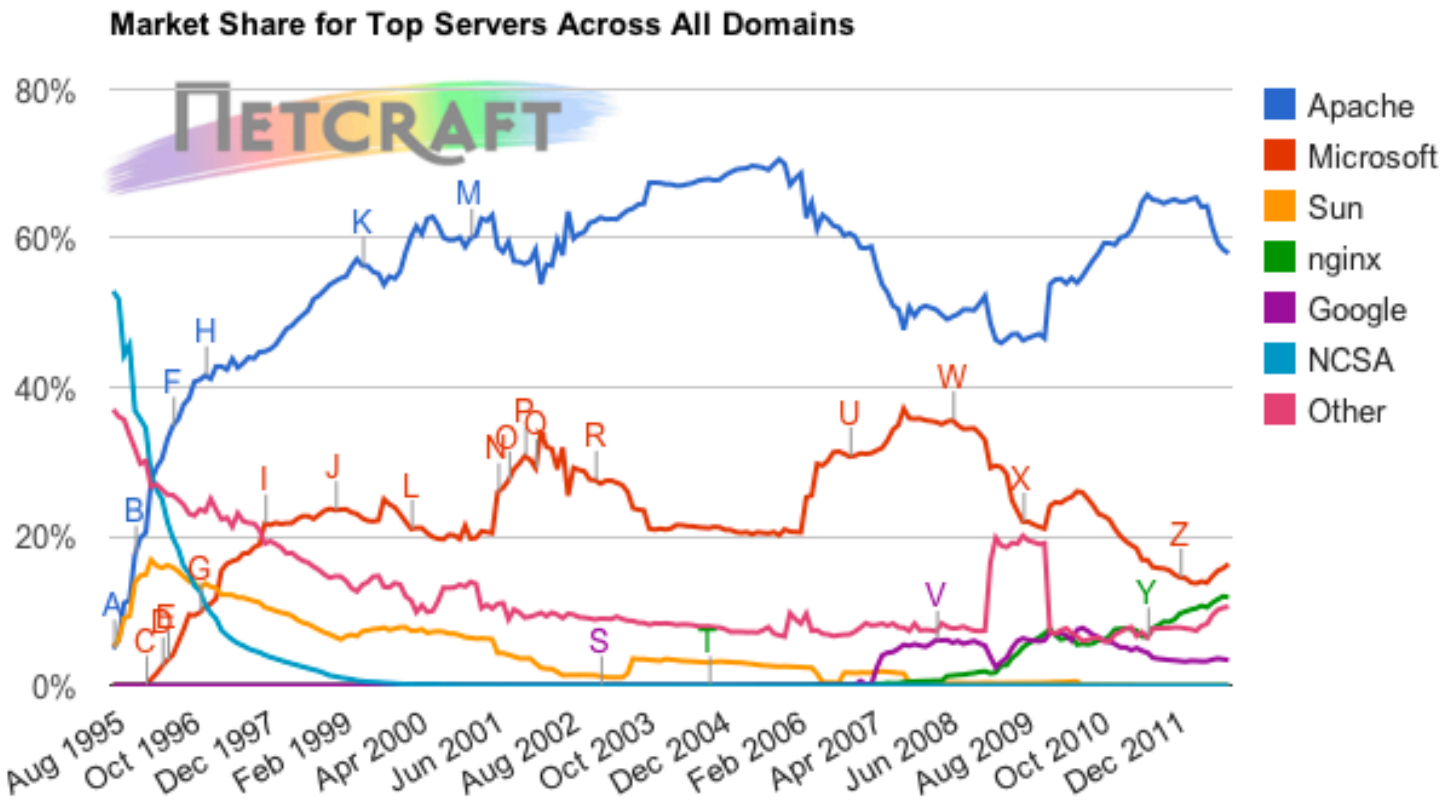
```
GET /index.html HTTP/1.0
Host: www.blah.com
User-agent: Mozilla/4.0
```

```
HTTP/1.0 200 OK

<html> …
```

Web server flavors

Server	Developed by	Open source	Software license
AOLserver	NaviSoft	Yes	Mozilla Public License
Apache HTTP Server	Apache Software Foundation	Yes	Apache License
Apache Tomcat	Apache Software Foundation	Yes	Apache License
Appweb	appwebserver.org	Yes	GPL and Commercial Versions
Boa	Paul Phillips	Yes	GPL
Caudium	The Caudium Group	Yes	GPL
Cherokee HTTP Server	Álvaro López Ortega	Yes	GPL
Hiawatha	Hugo Leisink	Yes	GPL
HFS	Rejetto	Yes	GPL
IBM HTTP Server	IBM	No	Non-free/proprietary
Internet Information Services (IIS)	Microsoft	No	Non-free/proprietary
Jetty	Eclipse Foundation	Yes	Apache License
KLone	KoanLogic Srl	Yes	GPLv2 and Commercial Version
lighttpd	Jan Kneschke (Incremental)	Yes	BSD variant
LiteSpeed Web Server	LiteSpeed Technologies	No	Non-free/proprietary
NaviServer	Various	Yes	MPL 1.1
NCSA HTTPd	Robert McCool	Yes	Free for Non-Commercial Use
nginx	Igor Sysoev	Yes	BSD variant
nodejs	Ryan Dahl	Yes	MIT
OpenLink Virtuoso	OpenLink Software	Yes	GPL and Commercial Versions
Oracle HTTP Server	Oracle Corporation	No	Non-free/proprietary
Oracle iPlanet Web Server	Sun Microsystems	Yes	BSD
Oracle WebLogic Server	Oracle Corporation (formerly BEA Systems)	No	Non-free/proprietary
thttpd	Jef Poskanzer for ACME Laboratories	Yes	BSD variant
Tornado	FriendFeed/Facebook	Yes	Apache License
TUX web server	Ingo Molnár	Yes	GPL
WEBrick	Ruby developers	Yes	Ruby license
Xitami	iMatix Corporation	Yes	GPL
Yaws	Claes Wikström	Yes	BSD variant
Zeus Web Server	Zeus Technology	No	Non-free/proprietary
Zope	Zope Corporation	Yes	ZPL

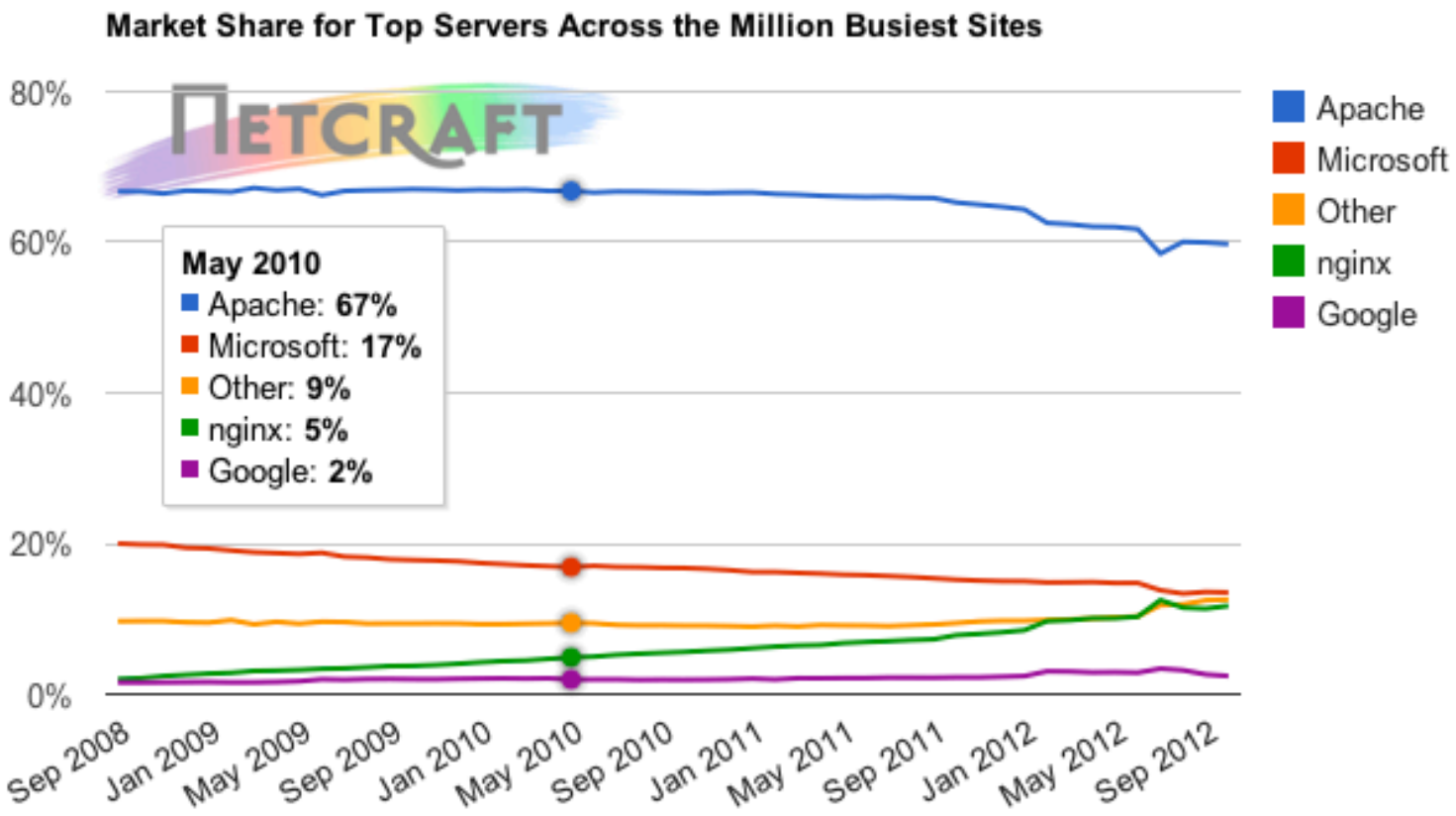
Web servers in use



Developer	September 2012	Percent	October 2012	Percent	Change
Apache	362,714,083	58.49%	359,875,516	58.00%	-0.49
Microsoft	97,368,803	15.70%	101,005,285	16.28%	0.58
nginx	73,976,009	11.93%	73,243,944	11.80%	-0.12
Google	21,576,233	3.48%	20,947,340	3.38%	-0.10

http://news.netcraft.com/archives/2012/10/02/october-2012-web-server-survey.html

Web servers in use



Developer	September 2012	Percent	October 2012	Percent	Change
Apache	596,589	59.98%	594,091	59.76%	-0.22
Microsoft	134,978	13.57%	134,319	13.51%	-0.06
nginx	112,991	11.36%	116,652	11.73%	0.37
Google	26,117	2.63%	24,047	2.42%	-0.21

http://news.netcraft.com/archives/2012/10/02/october-2012-web-server-survey.html

nginx???

- nginx
  - "engine-X"
    - Originally designed for Russian Rambler sites, 500M request/day
  - Free open-source
  - Designed for high performance, low memory footprint
    - C10K problem, 10000 simultaneous connections
  - Facebook, Zappos, Hulu, Dropbox, Wordpress…

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

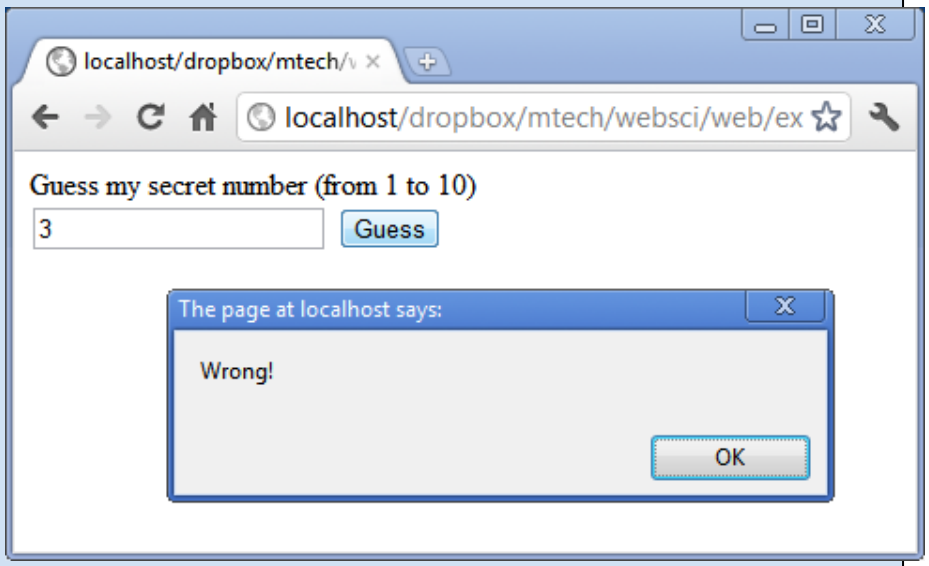
Static vs. dynamic

- Static content
  - Images and pages don't change
    - Always the same, like a file server
  - Fast to deliver, easy to cache, etc.
- Dynamic content
  - Same URL may result in different delivered HTML
    - e.g. different preference on # of products to display
  - May change as user interaction progresses
    - e.g. adding items to a shopping cart
  - Need something besides just HTTP and HTML
    - HTTP is stateless
    - HTTP not programmable (e.g. conditional, loops)

Scripting

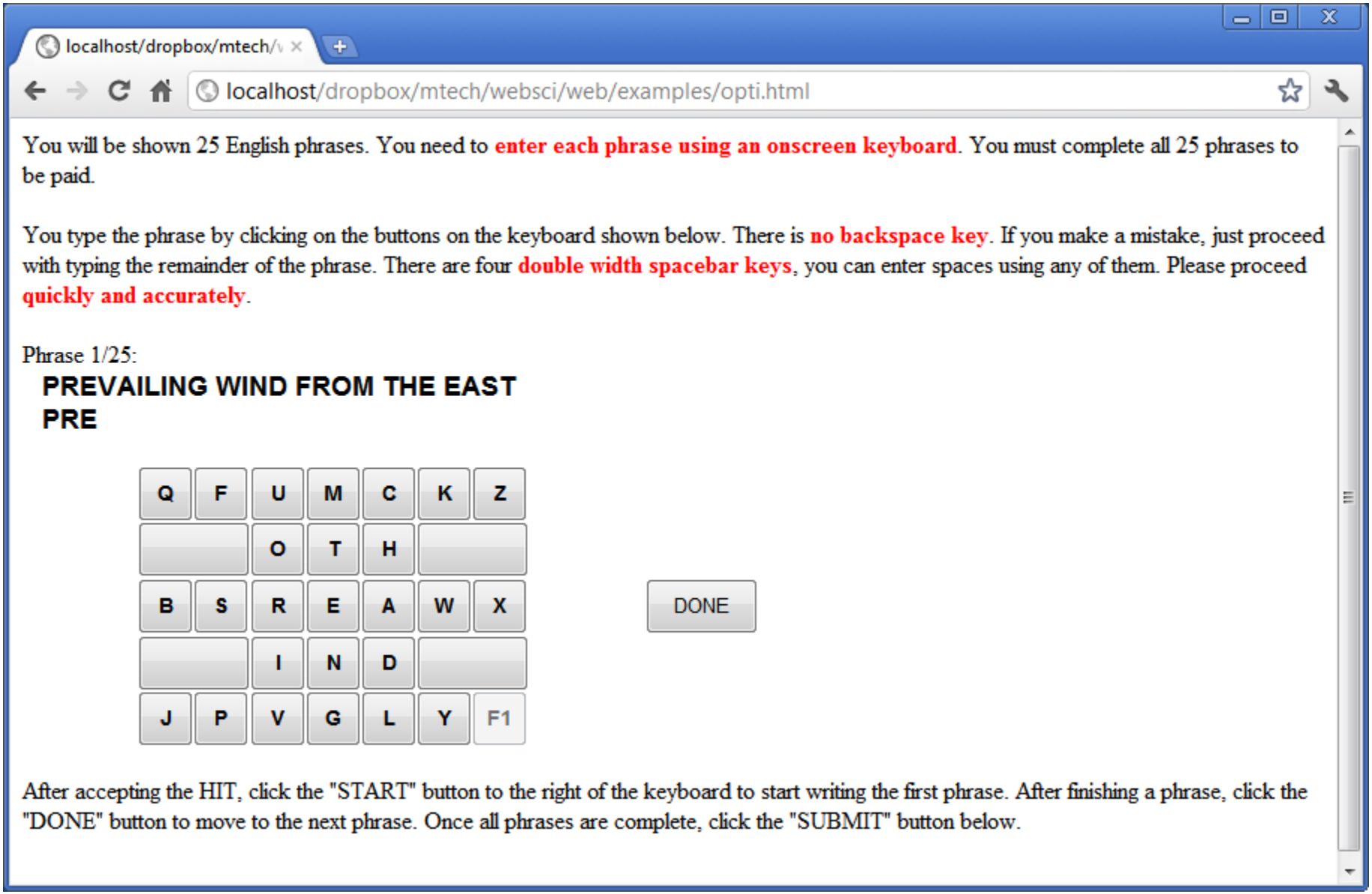
- Client-side scripting
  - Runs in the client's browser
    - Fast, no roundtrip to the server
  - Users can see if they "view source"
  - De facto client-side language: JavaScript
    - Heavily influenced by Java, but not the same
    - Weakly typed
  - The "dynamic" in DHTML
    - e.g. validate form 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 load on web server from lots of small requests
    - One big request
  - Can be difficult to debug
    - JavaScript console in browsers help
  - JavaScript tied to browser / client
    - Everything lost if browser is closed or crashes
    - 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
    - Repeated HTML (e.g. header/footer) in a single file

CGI

- CGI (Common Gateway Interface)
  - In use since 1993
  - URL request in a special location/file extension
    - e.g. http://www.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 web client
  - Notes:
    - Program must have read+execute permission by web server user
    - Probably shouldn't be writeable by anybody

CGI in Apache

```
...
<IfModule alias_module>
   ScriptAlias /cgi-bin/ "/usr/local/apache2/cgi-bin"
</IfModule>
...
```

httpd.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

```
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

```
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?foo=bar&hello=world

CGI in C

- CGI in C
  - Use getenv() function 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]
  - Switches:
    - -n number of requests
    - -c concurrent requests

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

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

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

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
  - Large variety with different tradeoffs, one choice: CGI
- Benchmarking using Apache benchmark, ab
- Possible paper topic:
  - What do high performance web servers such as nginx and lighttpd do differently from Apache?
    - Discuss feature differences, advantages, disadvantages
    - Empirical performance comparison