

Analysis + CGI

Finishing the discussion of network effect economics

Proprietary Picture

e.g. Nikon/Canon, each with their own little domain of compatibles lenses

e.g. GM car only works with GM gas

e.g. Cell phone

e.g. Printer ink cartridges

1. Consumer suffers

The vendor has created, in effect, a mini-monopoly

This is certainly worse for the consumer

2. Vendor suffers?

This is a more controversial point.

The vendors may also be worse off. If they worked with a standard lens mount, they would have to compete, but the market might be so much bigger that the vendors would still be better off.

e.g. GM sells more cars, because the gasoline market is so large and well developed.

Nikon's monopoly on lenses extracts a tax on its consumers that obviously makes Nikon richer. However, the tax may decrease the size of the market so that, in sum, Nikon is worse off.

Results

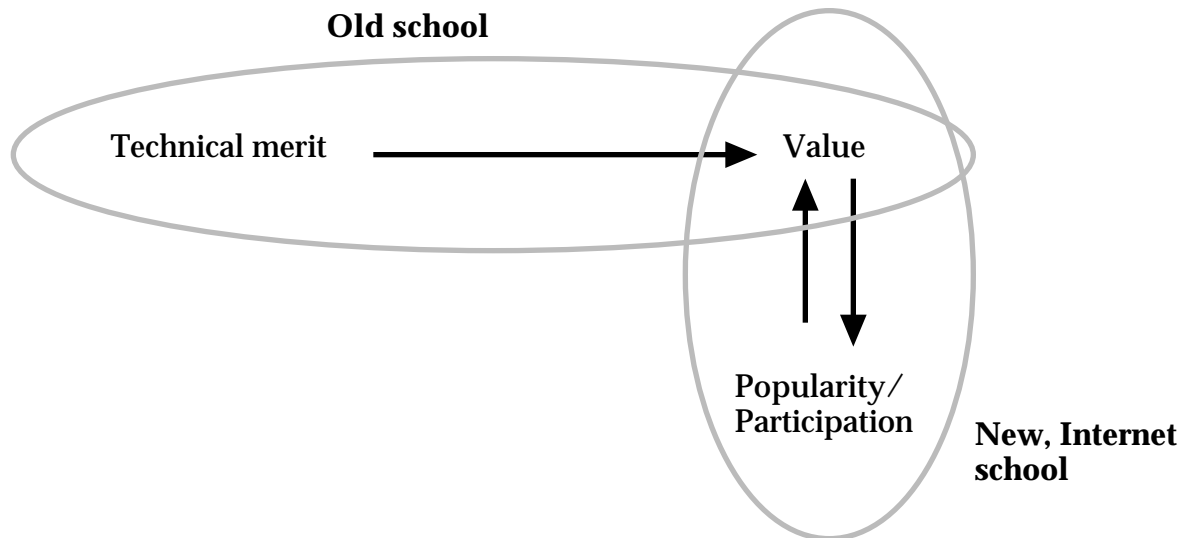
Be an intelligent consumer. Don't accept vendor lock-in without some thought.

Give the vendor a hard time about needlessly non-standard parts. You can use proprietary parts, but recognize that there is a cost to you, and to everyone.

If you are running a government agency in charge of, say, public radio frequencies, require that the frequency be used with a public standard technology (with low to zero patent barriers to implementation). This is certainly in the public's interest, although the vendors may fight it. (They want to defect; you make them cooperate with each other.)

Tech Merit vs. Network Effect

The core argument between the two schools of thought...



MS's rise to dominance is an example of proprietary network effect (aka inertia). The Internet's rise to dominance is an example of public standard network effect growing to an even greater dominance. (e.g. TCP/IP, HTTP, ...)

Q: Where did the Internet Come from?

What is its growth engine?

A: Standards

Any-Any Synergies

Value is unlocked by the any-to-any connections enabled by freely implementable standards.

Q: What Makes Standards Work?

e.g. SMTP and POP are nothing special -- how is it that they unlocked so much value so quickly and cheaply?

A: Voluntary Participation

The value is unlocked by the voluntary participation.

"It's the participants, stupid."

Without the voluntary participants, the system is worth nothing. As a result, the future depends on standards and incentives for participation more than technical merit. Public, non-proprietary standards are especially attractive to participants -- the Internet is a great example of this.

Q: What's so special about standards?

A:

Nobody knows for sure why standards work out so well. Here are some speculations

1. 100% much better than 90%

Maybe having 100% compatibility is much better than the 90% that a proprietary, e.g., MS, standard can reach. Maybe the 100% unlocks any-any synergies.

MS can never reach 100% because they are all about setting up tech tying (or we could say "compatibility") between products they control and deliberately making non-MS systems not work -- this always leaves them at 90% at most. Perhaps leaving out the last 10% hurts disproportionately.

2. Ego vs. proprietary standard

Maybe it's just unappealing to participate in a standard owned by someone else, so the critical mass of participation never quite blossoms. If you are writing some software tool that outputs documents -- are you going to reverse engineer the .doc format so you can produce it, or are you going to just write out HTML. A public, free standard like POP is more appealing to the participants since you're not obviously feathering some else's bed. Also, the proprietary standards do not have the documentation and history of backward compatibility. Quite the opposite -- it may be in Microsoft's interest to make sure that the .doc format changes every few years. Theme: incentives for participation matter more than technical merit

3. Competition / Darwinism

The standard makes implementations replaceable which leads to competition. One of the results over time is that the implementation price/quality gets better and better. This is what happened with the IDE/ATA disk standard. Over time, it got tech merit through evolution and competition, fueled by the huge and competitive market.

Q: Why didn't MS invent the Internet?

MS word has over 90% of the word-processing market share. Likewise, MS networking technology. Why didn't "The Internet" grow up around MS document interchange?

A:

See above -- apparently free standards pull in participants in a way that the MS standard doesn't quite.

MS is all about setting up synergies between its products and leaving out the products of others. Create synergies between MS products without helping or encouraging competition. From a sufficient position of strength, this gets the Network Effect working for MS's interests. All vendors would love to use this strategy given the chance.

What Would Bill Gates Say?

1. Standards Are Bad

Standards have slow, designed-by-committee feel
They must be dumbed down to the lowest common denominator

Response

That's all true, but technical merit was not the point.
Participation and standards-driven competition end up mattering more so long as the standard is of at least reasonable quality.

2. IE won on its merits

IE has more market share because it is better than Netscape.

Response

IE is a high quality implementation. IE is dominating with a combination of tech merit and an illegal monopoly tying strategy. (Had MS not "cheated" in this way, IE might still have won on its merits, but we'll never know.)
MS has also done every sort of technical and marketing tying/bundling imaginable to channel inertia and Network Effect MSs way as much as possible. MS understands inertia and its importance over technical merit.

3. Merit Beats Inertia

MS products have 90% market share because they are better than the competition.

Response

Merit is good, but inertia is better.
Since the DOS days, MS has well understood the role of inertia and network effects in product success. Gates was the first to really understand and exploit the dynamics of a high-inertia marketplace.
MS produces pretty good products, but the overall MS domination is a testament to proprietary network effect.
If MS thinks its a merit-driven world, why are they so hot to get IE up to 90% market share as soon as possible? It's because they realize that whoever dominates a niche first, dominates it forever because the network effect dominates tech merit.

Microsoft vs. IBM inertia quote

The paragraph below describes IBM's plan to collaborate with Microsoft to write IBM's OS/2 software. Ultimately, OS/2 was destroyed by Microsoft's own product in that niche, Windows, and it's obvious to see how. IBM's strategy

exactly shows that they do not understand the tech-merit vs. inertia dynamic, while Microsoft does...

<http://www.nytimes.com/library/tech/yr/mo/biztech/articles/18soft.html>

(this link no longer works, because the NYT does not understand that they should just take over the reference-URL marketplace while it is still young)

...In November 1989, when IBM and Microsoft were still collaborating on OS/2, **they designed it to run only on expensive computers with more memory than was typical at the time. Until hardware caught up, the companies said, customers could use Microsoft's new product, Windows.** Eventually, however, the computing world turned to Windows, and Microsoft ended its partnership with IBM.

Translation:

We'll let the customers use product A for a couple years. But then we'll switch them to product B which has more tech merit. This does not work: product A gets the network effect and inertia going, and then product B is irrelevant.

Server Side Example - Logs

Each HTTP request gets one line in the log:

Client-IP URL [date] "request" result-code byte-size "referer"

Requests

See how every .gif is its own little requests

See how the requests are each handled individually, even requests from multiple clients may interleave over the course of a few seconds.

Referer

The page that included the url that lead to this request

Often a cslibrary page pointing to cslibrary

Could be a google/yahoo page pointing to cslibrary

Result Codes

200 -- success

206 -- IE bug where it requests binary files, like PDF, multiple times. A cynical person would say that MS doesn't mind this sort of bug -- where MS software interacts badly with the HTTP standard, but pehaps works well with the MS HTTP server. I think it's probably just a bug.

301 -- redirect, probably for missing /

304 -- the client cache version is up to date

Other

The "favico.ico" is a neat, but proprietary idea. It's the little icon that goes next to the bookmark in IE when you bookmark a page. There should be a standard for this, and it should use a standard image format such as PNG.

```
61.141.93.68 - - [09/May/2001:07:14:09 -0700] "GET / HTTP/1.1" 200 7376
"http://cse.stanford.edu/"
```

Link to / from cse.stanford.edu

```
61.141.93.68 - - [09/May/2001:07:14:11 -0700] "GET /smallbinky.gif HTTP/1.1"
200 7648 "http://cslibrary.stanford.edu/"
```

Image on that page

```
61.141.93.68 - - [09/May/2001:07:14:32 -0700] "GET /107/ HTTP/1.1" 200 1545
"http://cslibrary.stanford.edu/"
```

```
61.141.93.68 - - [09/May/2001:07:14:41 -0700] "GET
/107/UnixProgrammingTools.pdf HTTP/1.1" 200 32768
"http://cslibrary.stanford.edu/107/"
```

Click to page 107, download the PDF there

```
61.141.93.68 - - [09/May/2001:07:14:44 -0700] "GET
/107/UnixProgrammingTools.pdf HTTP/1.1" 206 32768 "-"
```

```
61.141.93.68 - - [09/May/2001:07:14:51 -0700] "GET
/107/UnixProgrammingTools.pdf HTTP/1.1" 206 32768 "-"
```

```
61.141.93.68 - - [09/May/2001:07:14:52 -0700] "GET
/107/UnixProgrammingTools.pdf HTTP/1.1" 206 36513 "-"
```

61.141.93.68 -- [09/May/2001:07:14:53 -0700] "GET
/107/UnixProgrammingTools.pdf HTTP/1.1" 206 35685 "-"
IE bug

61.141.93.68 -- [09/May/2001:07:15:37 -0700] "GET /101/ HTTP/1.1" 200 1954
"http://cslibrary.stanford.edu/"
61.141.93.68 -- [09/May/2001:07:15:44 -0700] "GET /101/EssentialC.pdf
HTTP/1.1" 200 32768 "http://cslibrary.stanford.edu/101/"
61.141.93.68 -- [09/May/2001:07:15:46 -0700] "GET /101/EssentialC.pdf
HTTP/1.1" 206 32768 "-"
61.141.93.68 -- [09/May/2001:07:15:47 -0700] "GET /101/EssentialC.pdf
HTTP/1.1" 206 32768 "-"
61.141.93.68 -- [09/May/2001:07:15:48 -0700] "GET /101/EssentialC.pdf
HTTP/1.1" 206 32768 "-"
61.141.93.68 -- [09/May/2001:07:16:02 -0700] "GET /101/EssentialC.pdf
HTTP/1.1" 206 83546 "-"
206.98.43.52 -- [09/May/2001:07:16:44 -0700] "GET /103/ HTTP/1.1" 304 - "-"
206.98.43.52 -- [09/May/2001:07:16:49 -0700] "GET /104/ HTTP/1.1" 304 -
"http://cslibrary.stanford.edu/103/"
206.98.43.52 -- [09/May/2001:07:16:49 -0700] "GET /104/42.gif HTTP/1.1" 304 -
"http://cslibrary.stanford.edu/104/"
206.98.43.52 -- [09/May/2001:07:16:51 -0700] "GET /104/assign.gif HTTP/1.1"
304 - "http://cslibrary.stanford.edu/104/"

This client already has those in cache -- 304 result based on mod-date

192.25.169.65 -- [09/May/2001:07:35:10 -0700] "GET / HTTP/1.0" 200 7376 "-"
63.68.16.100 -- [09/May/2001:07:35:34 -0700] "GET /103/ HTTP/1.0" 200 2633
"http://www.google.com/search?q=linked+list"
A google search page pointing to /103/

193.170.244.2 -- [09/May/2001:07:37:07 -0700] "GET /104/ HTTP/1.0" 200 7314
"http://www.google.com/search?hl=de&safe=off&q=fun+video&lr="

193.170.244.2 -- [09/May/2001:07:37:10 -0700] "GET /104/assign.gif HTTP/1.0"
200 20975 "http://cslibrary.stanford.edu/104/"

193.170.244.2 -- [09/May/2001:07:37:10 -0700] "GET /104/42.gif HTTP/1.0" 200
20671 "http://cslibrary.stanford.edu/104/"

206.98.43.52 -- [09/May/2001:07:43:30 -0700] "GET /103/ HTTP/1.1" 304 - "-"
206.98.43.52 -- [09/May/2001:07:43:42 -0700] "GET / HTTP/1.1" 200 7376
"http://cslibrary.stanford.edu/103/"

Notice how the 206.98 requests are mixed in with the 192.25 and 63.68 requests -- the requests are jumbled together and handled one by one.

<snip>

194.74.228.16 -- [09/May/2001:07:57:20 -0700] "GET /105/favicon.ico HTTP/1.0"
404 289 "-"

MS proprietary "favicon.ico" thing -- little image when you bookmark.
This is a neat idea, but why not use a standard image format like PNG? .ico is a
MS only image format. Typical.

Server Side HTTP

HTTP server sits there

request comes in

Typically HTTP server maps that request into its file system

The server decides what the MIME type is -- probably based on its file extension.

This "file system" model is sometimes called "static pages"

Alternative Server Side - Dynamic

Suppose the server is just a database of classes "cs106, cs107, ..."

When a request comes in like "/class/cs106"...

1) Look up that row in the database

2) Dynamically generate a page containing that row's data and send it back as a reply

The client just specifies a request string such as "/" or "/class/cs106"

How the server interprets the request is ultimately up to the server

CGI

Common Gateway Interface -- a standard which interfaces the HTTP server software with CGI programs which run on the server.

The CGI standard defines how the server communicates all the various facts about the requesting URL (date, referer, accept) to the CGI program, and how the CGI program's output should be handled and sent back to the requesting client.

CGI is language independent: C, Perl, Unix shell script.

Security

Security concerns -- because the CGI's run on the server, they need to take care not to allow a rogue client to compromise the server either by hurting the server or serving up information which was supposed to be secret.

CGI Dialog

1. URL

The client has a URL as usual. However, the URL identifies not a page of HTML, but a CGI program on the server.

e.g. `http://www-cs-faculty.stanford.edu/cgi-bin/nick/hello.pl`

2. Client GET

The client does a GET request as usual

3. Server -> CGI

The server looks at the GET request, realizes it's a CGI.

4. Run CGI

Server runs the CGI program, feeding it the GET request as input. The GET information is fed to the CGI through environment variables.

5. CGI program

The CGI looks at the input and computes what it wants to compute.

6. HTTP Header

First, the CGI prints the HTTP header fields it wants, followed by a blank line.

7. HTML content

Then the CGI prints the body content it wants and then exits

8. HTTP Server

The HTTP server gathers the output of the CGI, adds more fields to the header, and sends it along to the client.

Trivial Example — hello.pl

```
#!/usr/bin/perl -w

## Hello.pl -- demonstrate a trivial CGI that prints
## out some HTML and the current time on this server.

use strict 'vars';

my($EOL) = "\015\012";

## This is a human-readable str of the current time
my($nowStr);
$nowStr = localtime();

## This line must be included in the header
print "Content-type: text/html$EOL$EOL";

## Write out the HTML content
print "<html><head><title>Hello.pl</title></head>\n";
print "<body>\n";
print "<h1>Hello.pl</h1>\n";
print "Hello there from CGI-land. It's currently '$nowStr'\n";
print "</body></html>\n";
```



hello.pl CGI Process

This CGI doesn't look at any input, it just produces an HTTP header followed by the same HTML body every time.

For its HTTP header, it just writes the `"content-type: text/html"` line followed by a blank line. The HTTP server fixes up the rest of the header on its way out to the client.

Input to CGI programs

Most likely, the CGI runs as a user called "WWW" or "CGI" which has limited read/write permission on the system. This helps limit some of the security danger, even if the CGI is poorly written.

Before the CGI is run, its environment variables are set to communicate the request — the primitive Unix method for programs to communicate with each other.

The `QUERY_STRING` environment variable will be the text after the `?` in the URL. There are many other environment variables set to indicate other things about the request -- see the `DumpEnv` example below.

Shell Script CGI

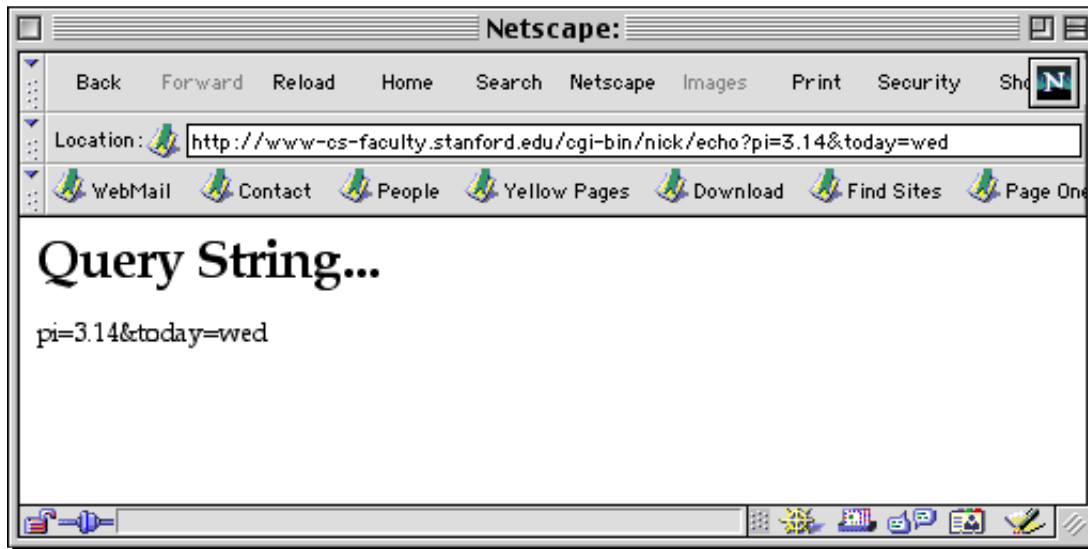
Shell script is a simple Unix programming facility which pre-dates the popularity of Perl. Environment variables are available by prefixing their name with a `$`. Shell scripts are a poor choice for CGI programming because they are easy to write in a way which creates a security hole -- allowing the client to run an arbitrary expression such as `"rm -rf *"` on the server.

Echo Query Shell Script

```
#!/bin/sh
# echo cgi script
echo "Content-type: text/html"
echo
echo "<html><body bgcolor=white>"
echo "<h1>Query String...</h1>"
echo "$QUERY_STRING"
echo "</body></html>"
echo
# Uncomment the following line for the definition of a security hole...
# $QUERY_STRING
```

Echo Query Example

Invoke: `http://www-cs-faculty.Stanford.EDU/cgi-bin/nick/echo?pi=3.14hello+there`



Usually, the query string encodes bindings with the "form" syntax:
`"n=10&pi=3.14"`

DumpEnv Perl Script

```
#!/usr/bin/perl
# Print out the values of all the environment variables
# in an HTML <ul>.
# Call from the shell or invoke as a CGI script.

# HTTP header section
print "content-type: text/html\r\n\r\n";

$header = <<EOT;
<html>
<head><title>DumpEnv</title></head>
<body bgcolor=white>
EOT

$trailer = <<EOT;
</body>
</html>
EOT

# Emit an HTML <ul> for all the environment vars
# set up for the CGI
print $header
```

```

print "<ul>\n";
## iterate over the keys, but sort them first
foreach $key (sort (keys %ENV)) {
    print "<li><b>$key</b> = $ENV{$key}\n";
}
print "</ul>\n";
print $trailer;

```

Environment Vars

Query_String (after the ? in the URL)

Script_Name (eg /cgi-bin/nick/dumpenv.pl)

Request_Method (usually GET)

Path_Info (extra path after the name of the CGI in the URL)

Remote_Host -- The client's IP or DNS address -- from a privacy point of view, this is one thing the server will definitely know -- how else will it send the packets back to you but by knowing your IP addr?

DumpEnv Output w/ "..?pi=3.14"

