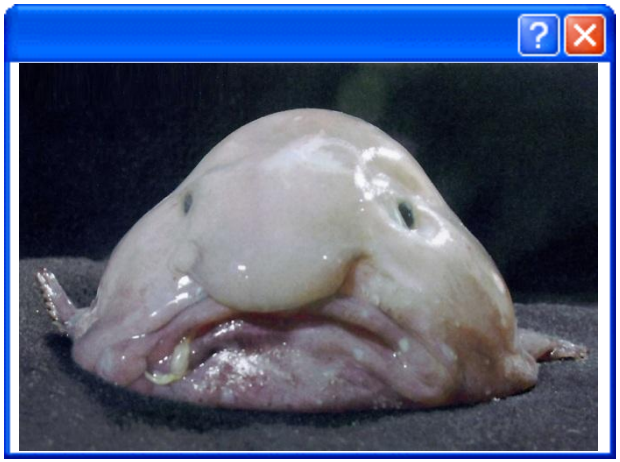
Embassies: Radically refactoring the web

John R. Douceur
Jon Howell
Bryan Parno
Microsoft Research

promise of the web model



the web is quite vulnerable

Buffer overflows

JavaScript API vulnerabilities

XSS

CSRF

Session fixation

clickjacking

safe web-surfing hygiene?

"don't click dangerous links!"

the problem

Security weaknesses in the web API

- complex execution semantics
- subtle communication & sharing semantics
- communication implicit in execution

cannot be fixed with a better browser for the same API

this talk

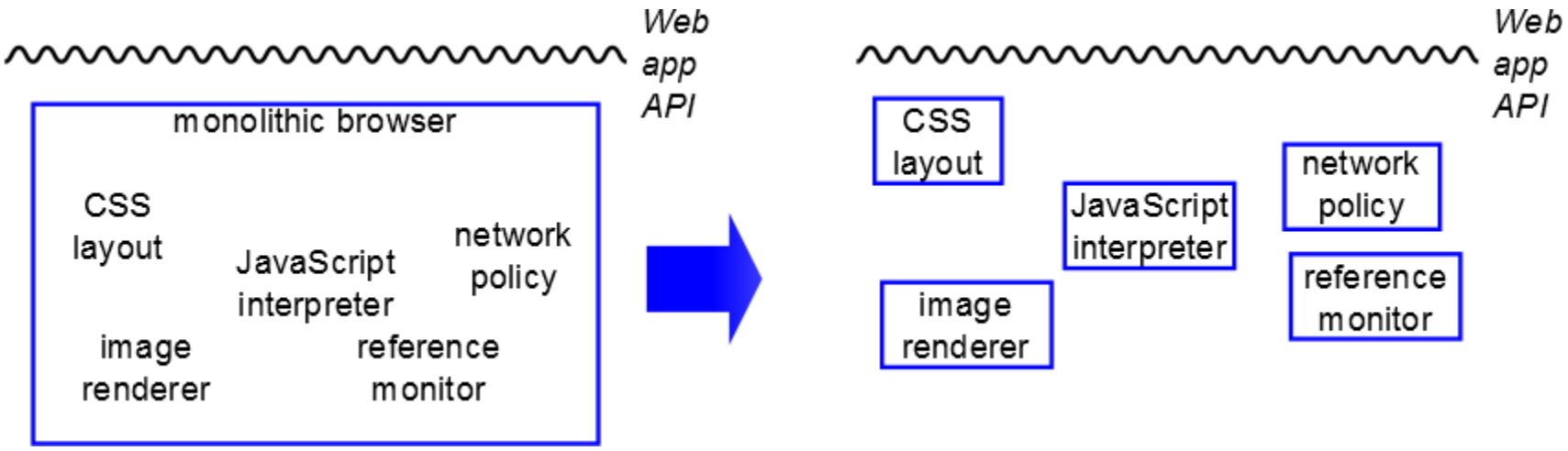
The current API is broken due to conflicting goals

Propose a new API for the web

- simple execution semantics: binary code
- explicit communication semantics: IP
- supports existing web apps and beyond

Argue that the new API evolves safely

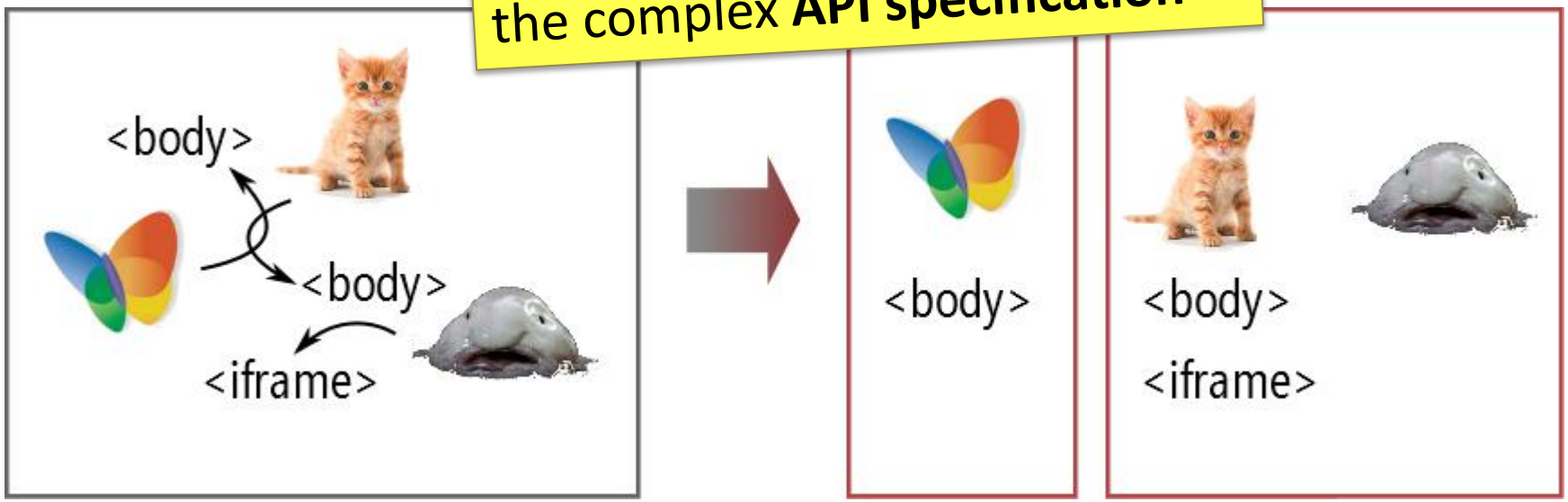
refactoring the browser isn't enough



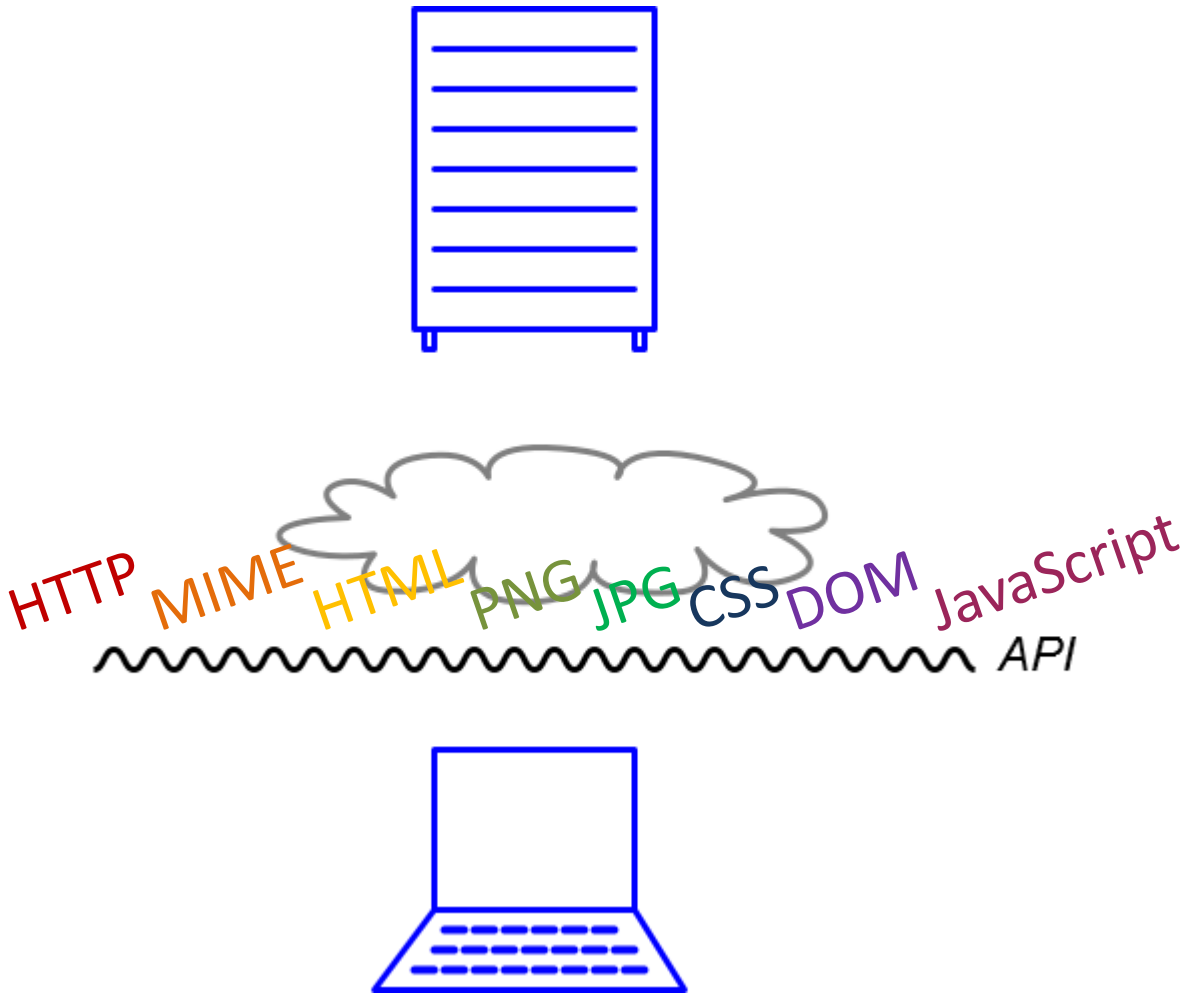
[OP, IBOS]

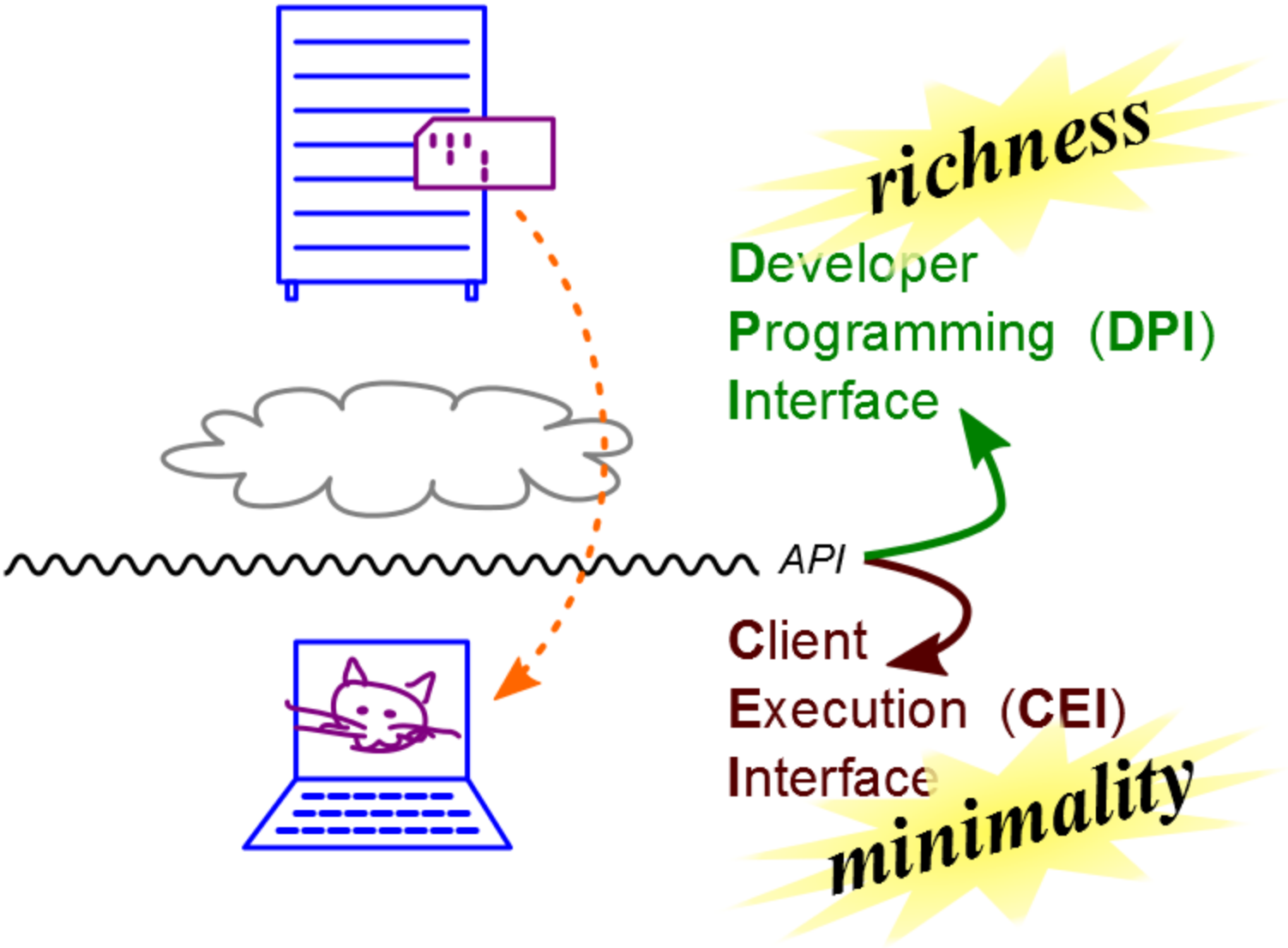
refactoring the browser isn't enough

the **browser implementation** doesn't matter until we fix the complex **API specification**

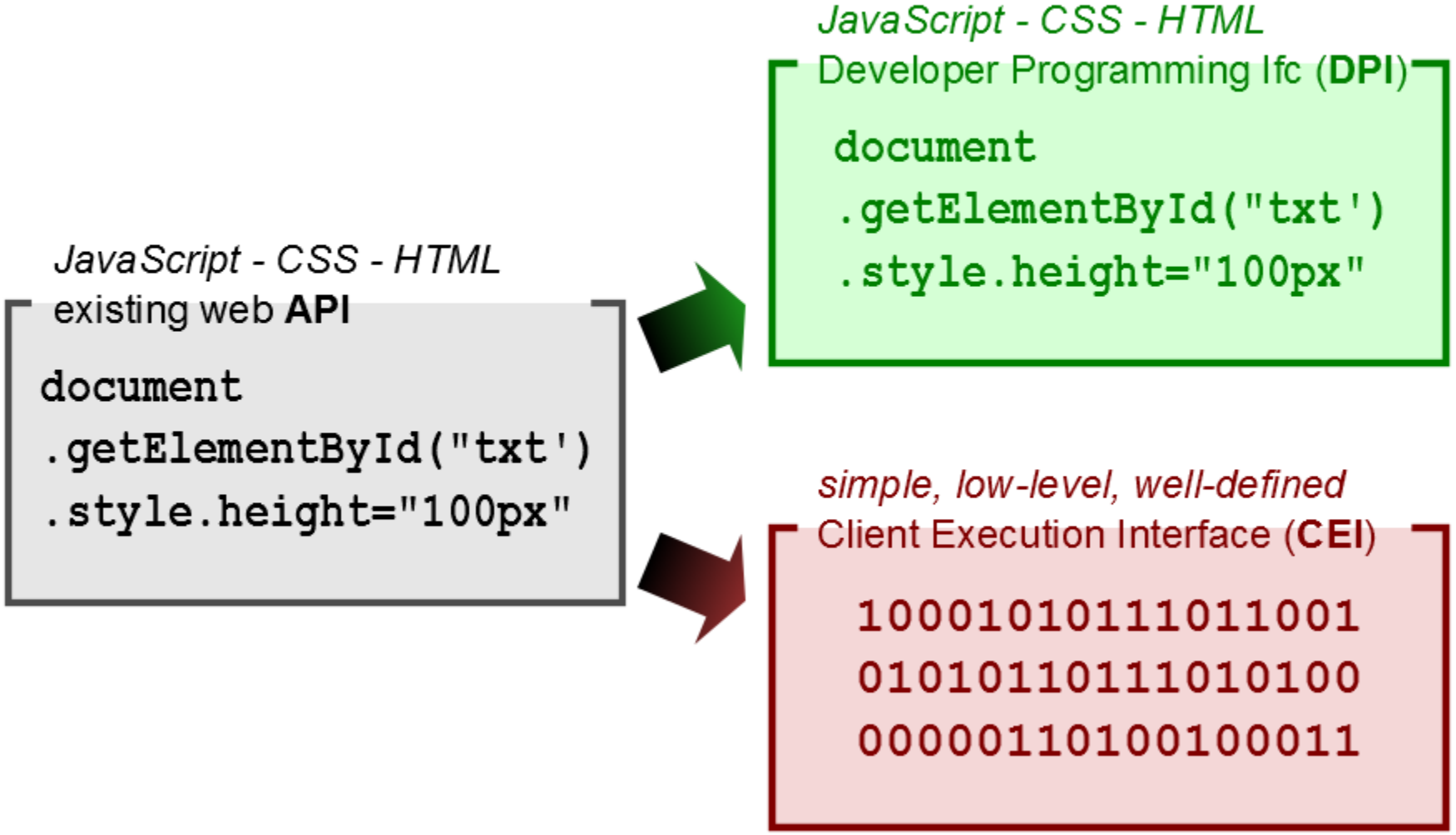


[Gazelle, Chrome]


HTTP
MIME
HTML
PNG
JPG
CSS
DOM
JavaScript
API

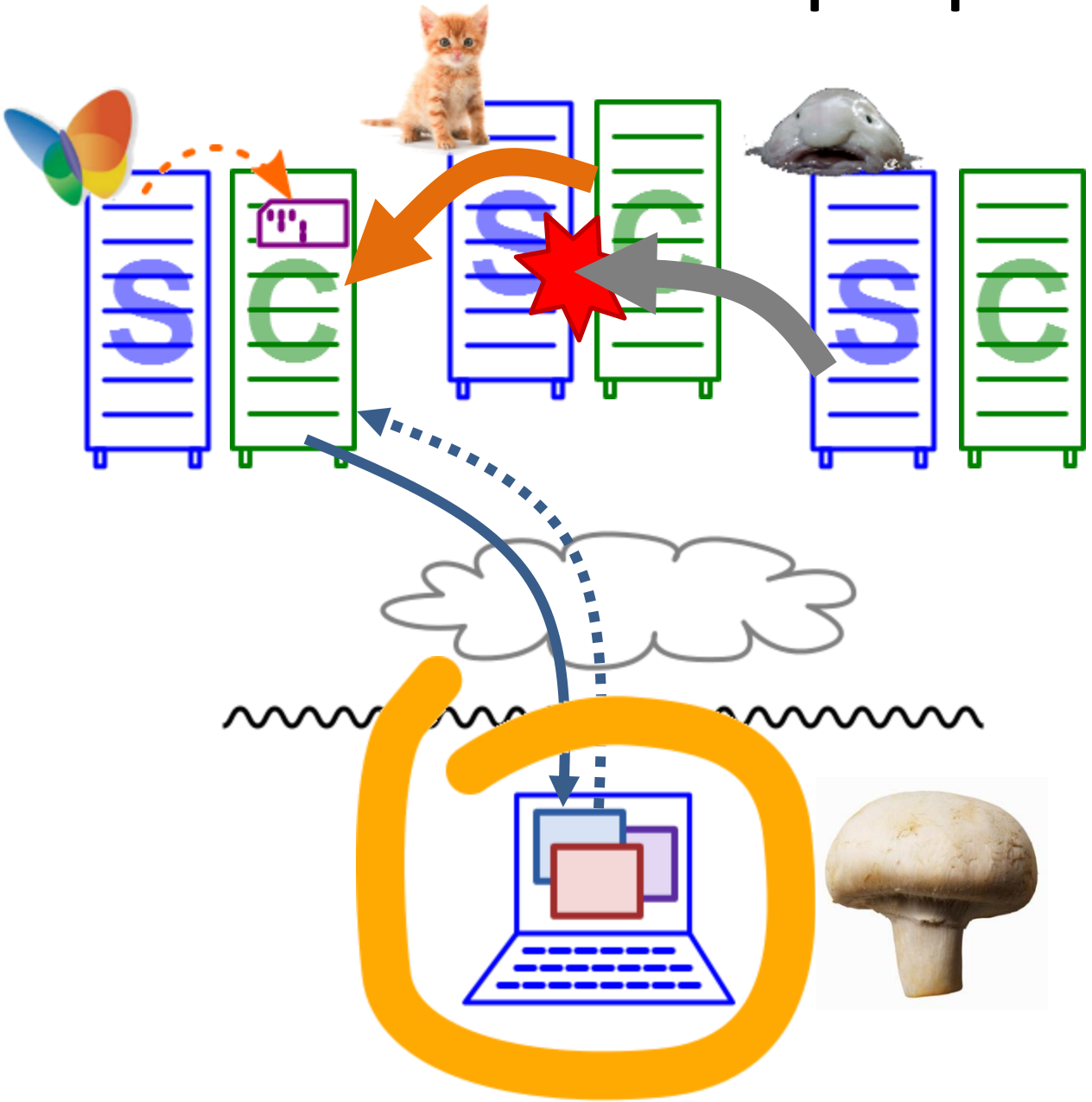

richness
Developer
Programming (DPI)
Interface
API
Client
Execution (CEI)
Interface
minimality

separate DPI from CEI



why is this model different?

a ridiculous straw-proposal



confounded by reality

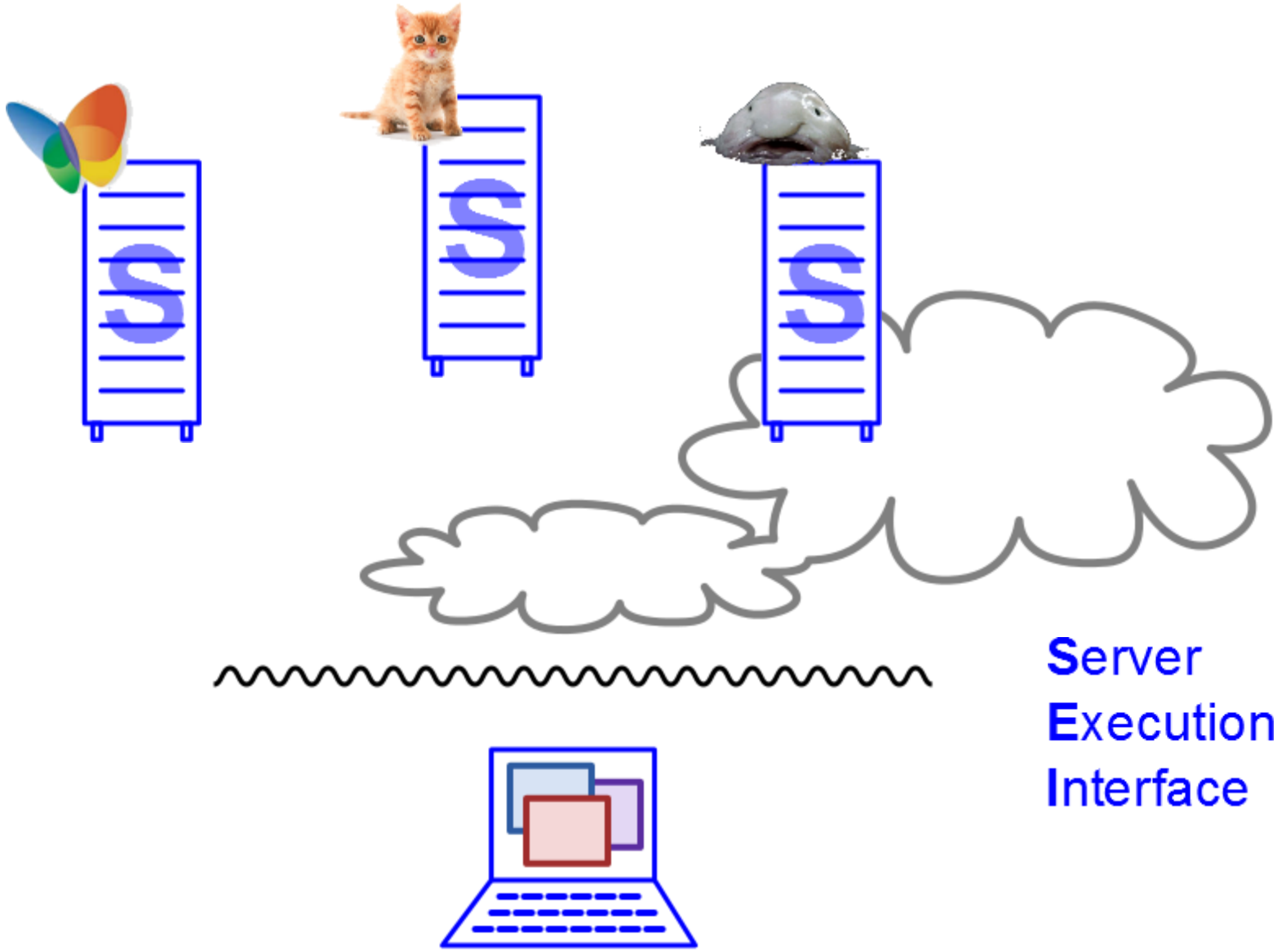
Network reliability

High bandwidth

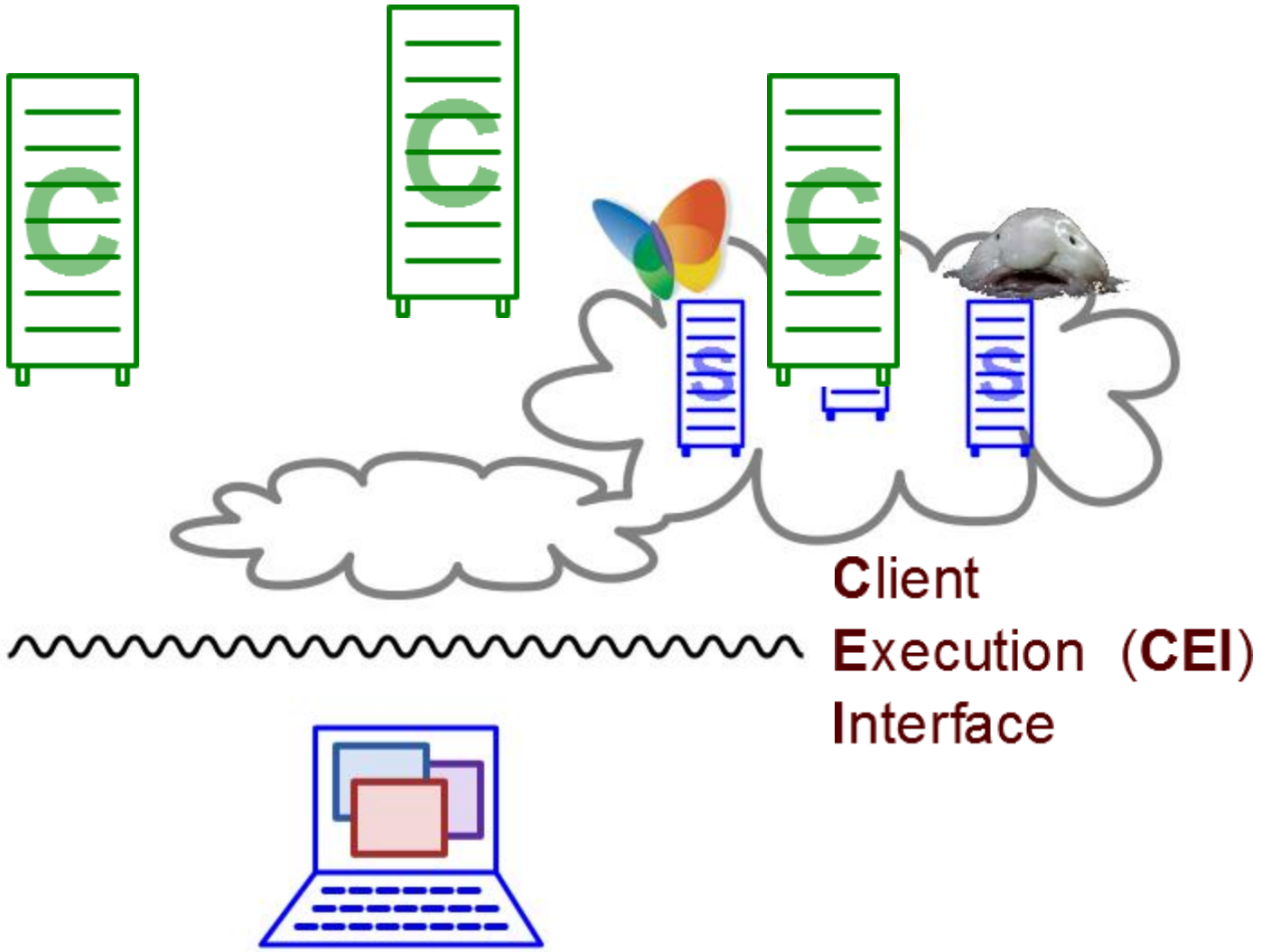
Low latency

Ample server resources

the multitenant datacenter



the client pico-datacenter



the entire Embassies CEI

execution: binary code

```
alloc_mem, free_mem
thread_create, thread_exit
x86_set_segments, exit
ensure_alive
futex_wait, futex_wake, get_alarms,
set_clock_alarm, get_time
```

communication: IP packets

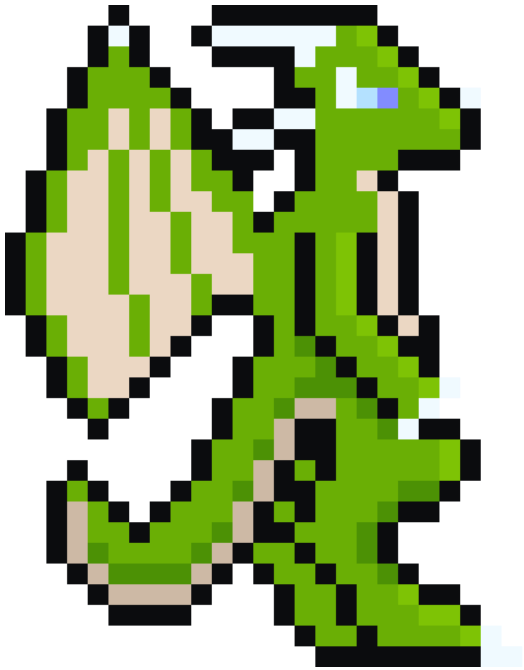
```
alloc_buffer, send_buffer
receive_buffer, free_buffer
```

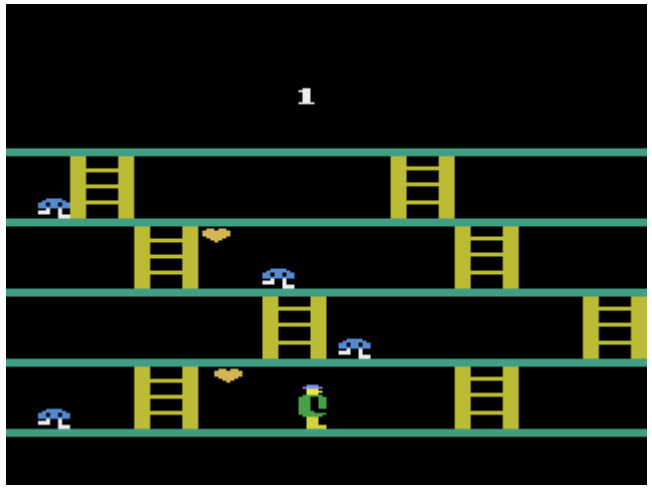
user interface: pixels and clicks

```
sublet_viewport, repossess_viewport
get_deed_key
accept_viewport, transfer_viewport
map_canvas, unmap_canvas, update_canvas
receive_ui_event
```

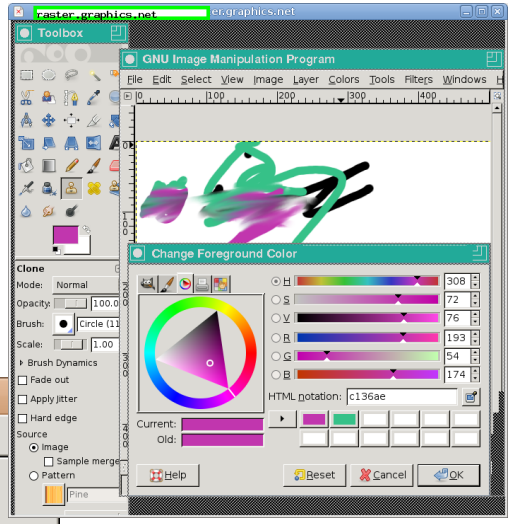
privacy & integrity primitives:

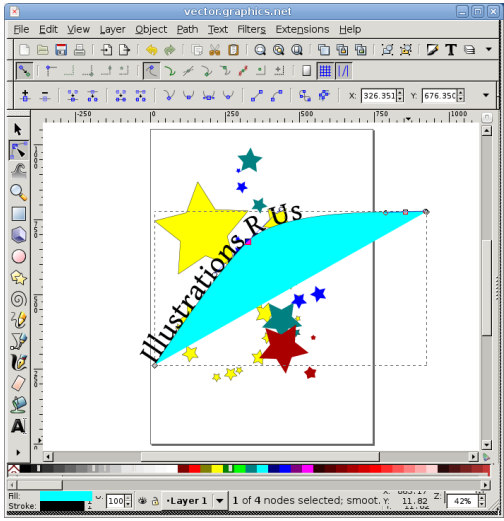
```
get_random, get_app_secret
endorse_me, verify_label
```

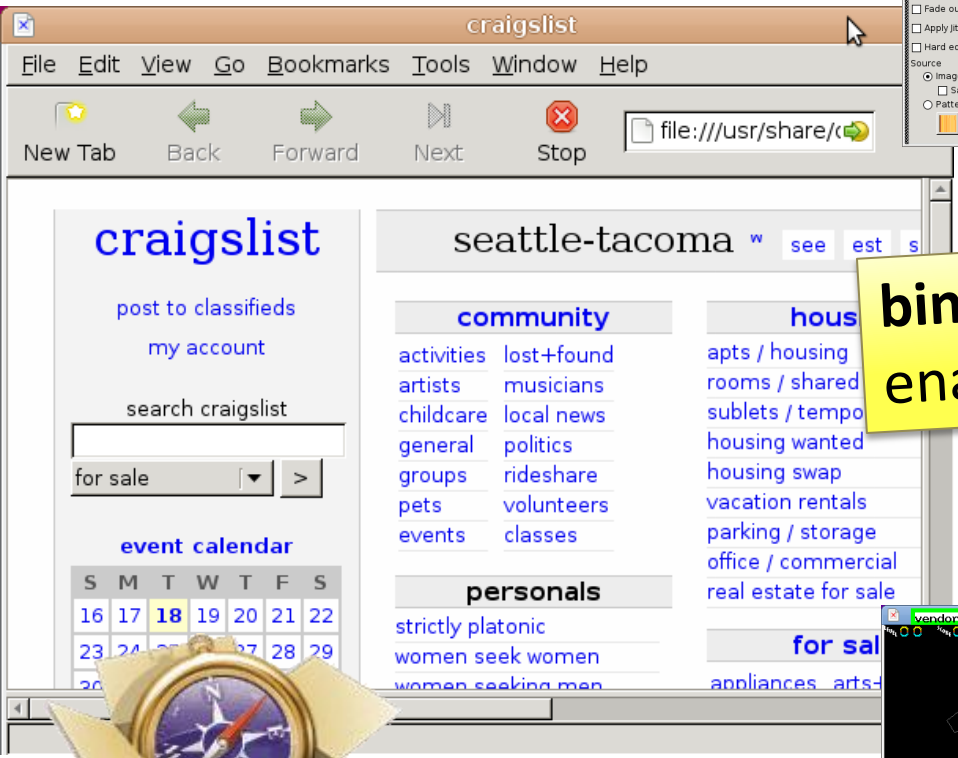



1

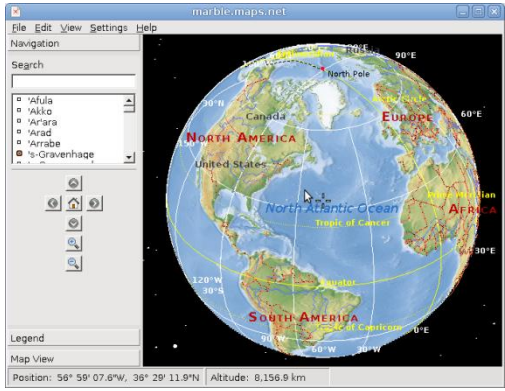
Developer Programming (DPI) Interface

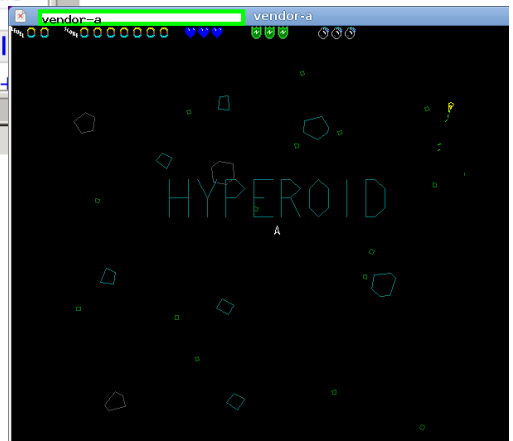




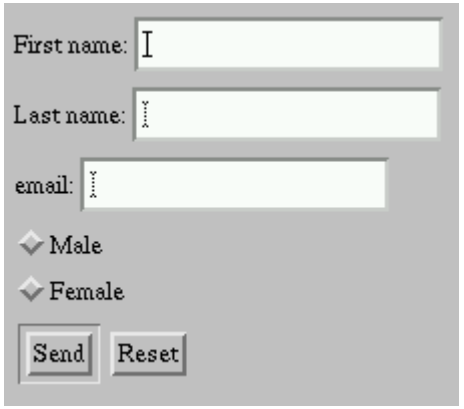


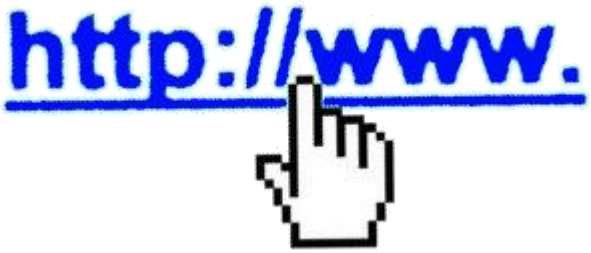
binary code
enables rich apps





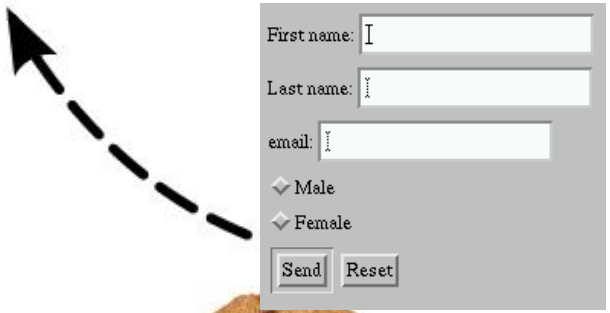
challenge: cross-app interactions



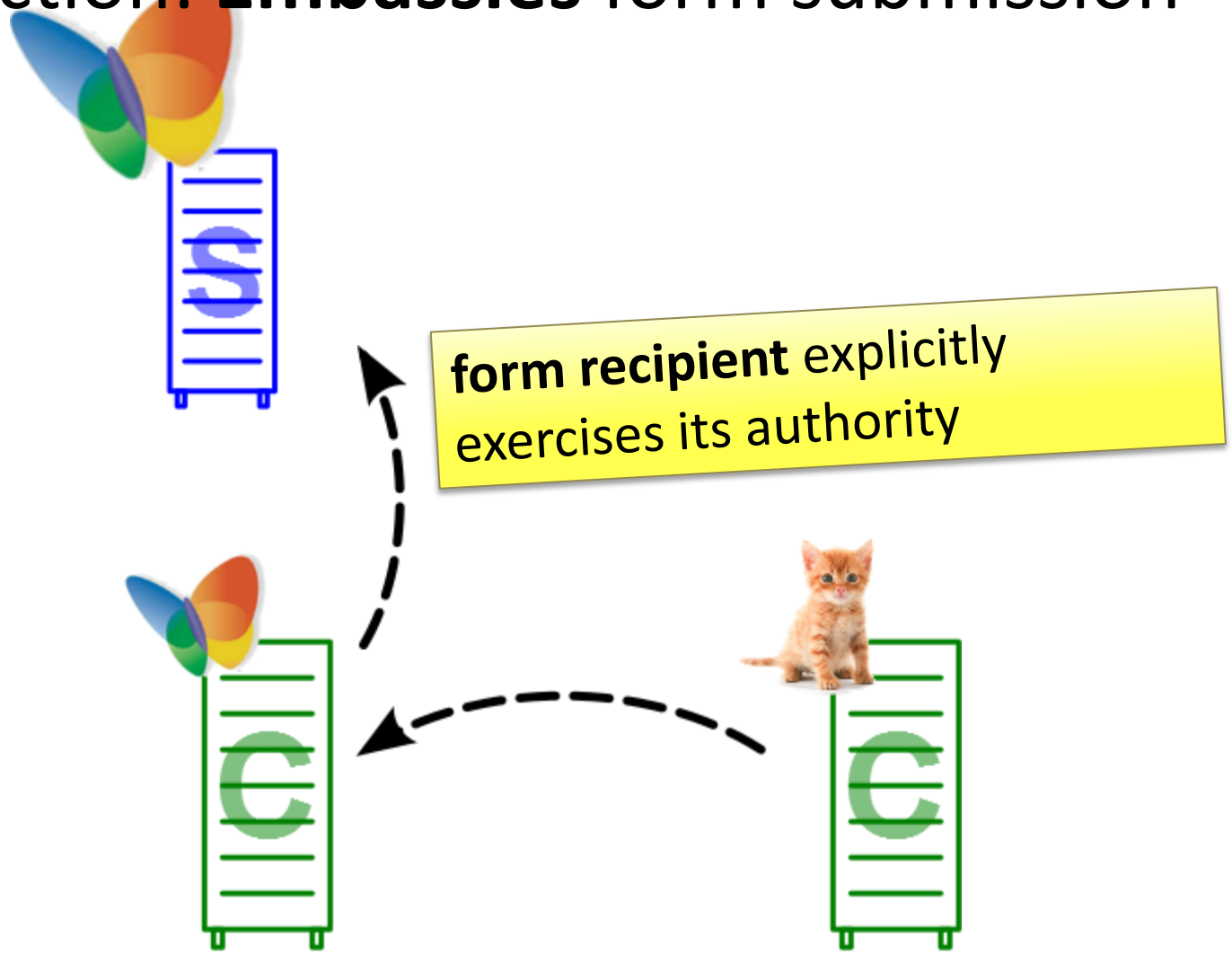


interaction: **today's** form submission

implicit ambient authority



interaction: **Embassies** form submission



interaction: **today's** link coloring

kittens.com/siamese

kittens.com/tabbies

kittens.com/calico

kittens.com/persian

kittens.com/siamese

kittens.com/tabbies

kittens.com/calico

kittens.com/persian

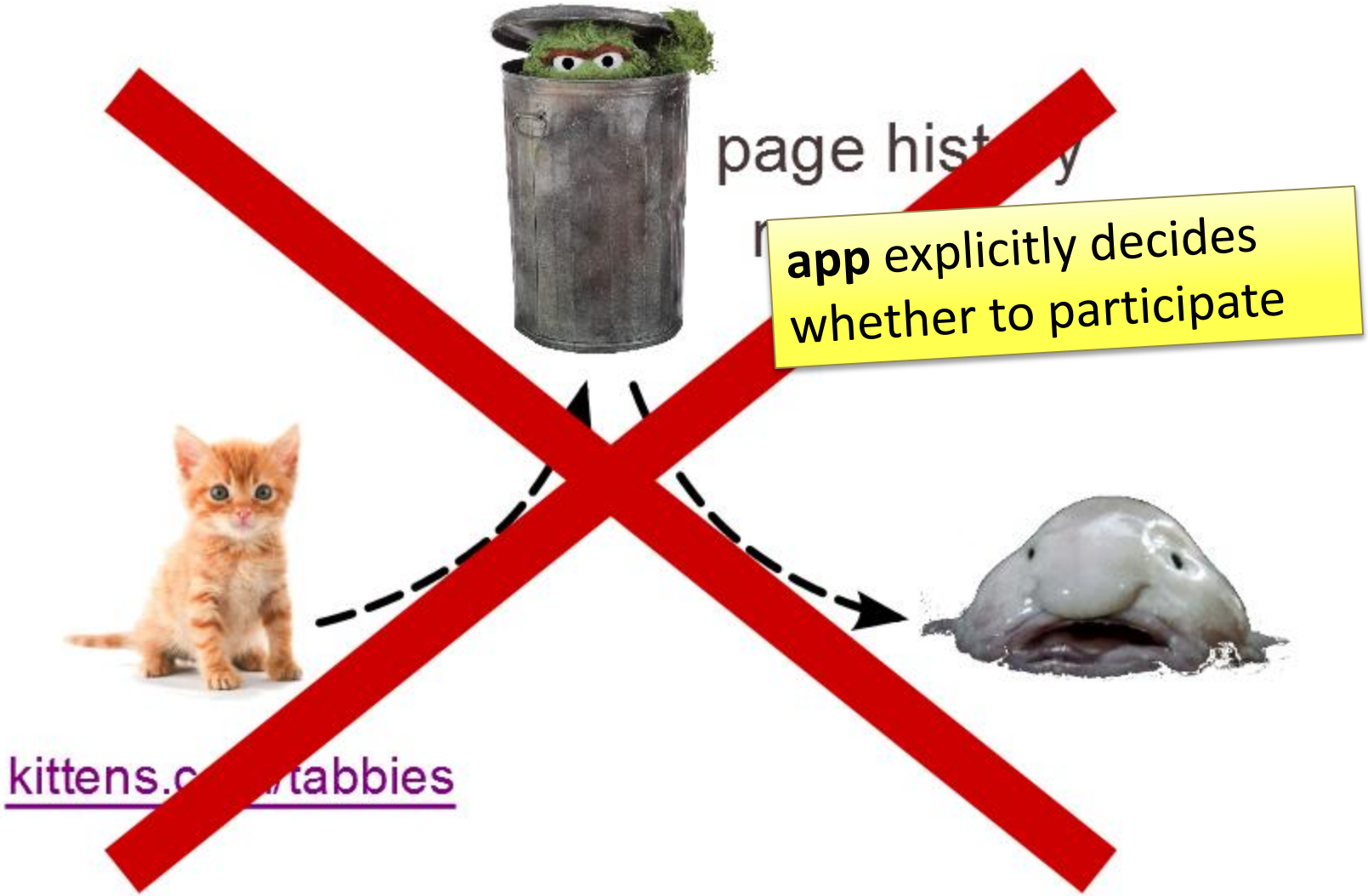
implicit history leaks

interaction: **today's** link coloring

page history repository

kittens.com/tabbies

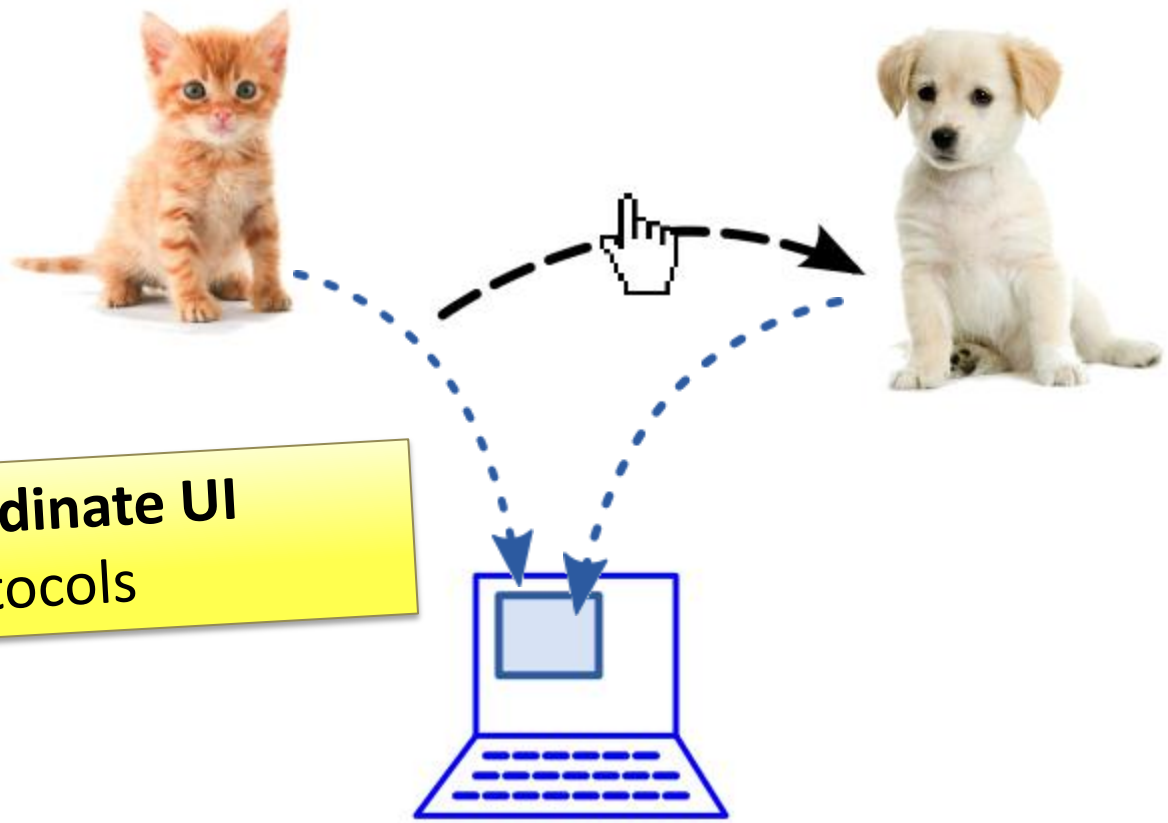
interaction: **Embassies** link coloring



interaction: **today's** page navigation

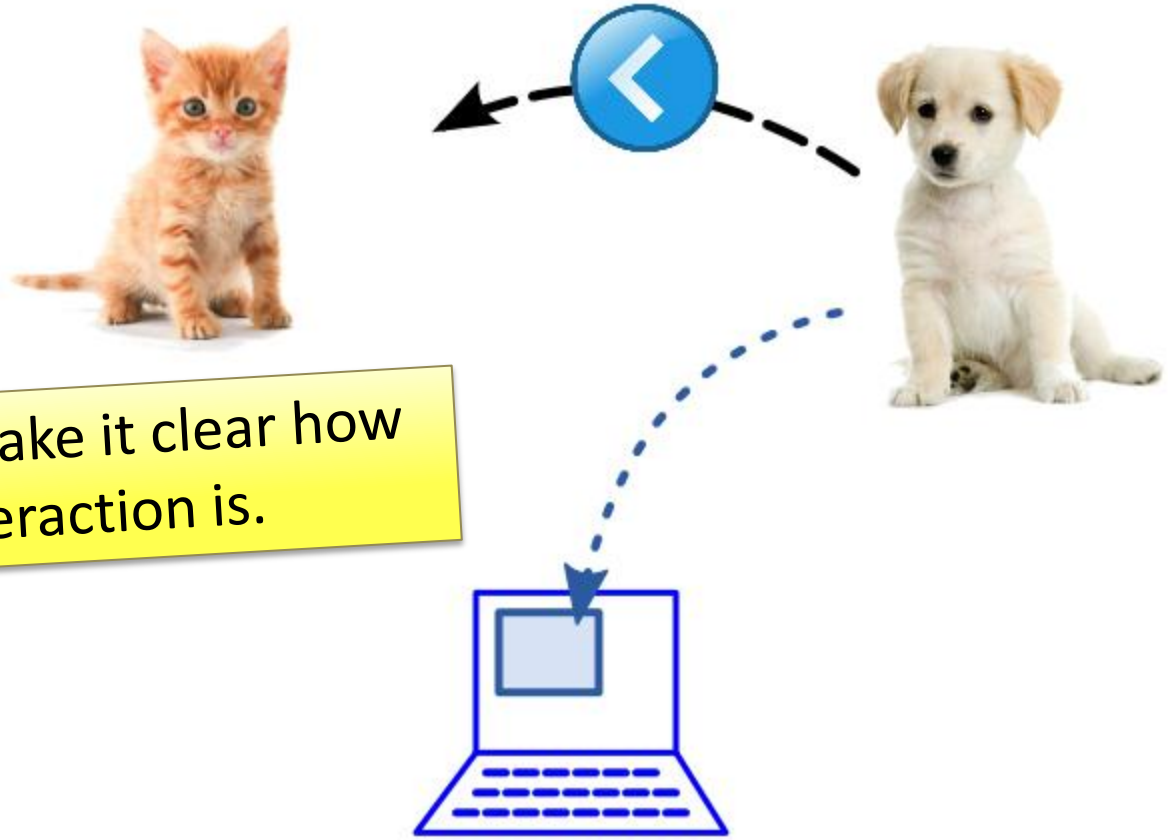
existing API requires browser to mediate navigation

interaction: **Embassies** page navigation



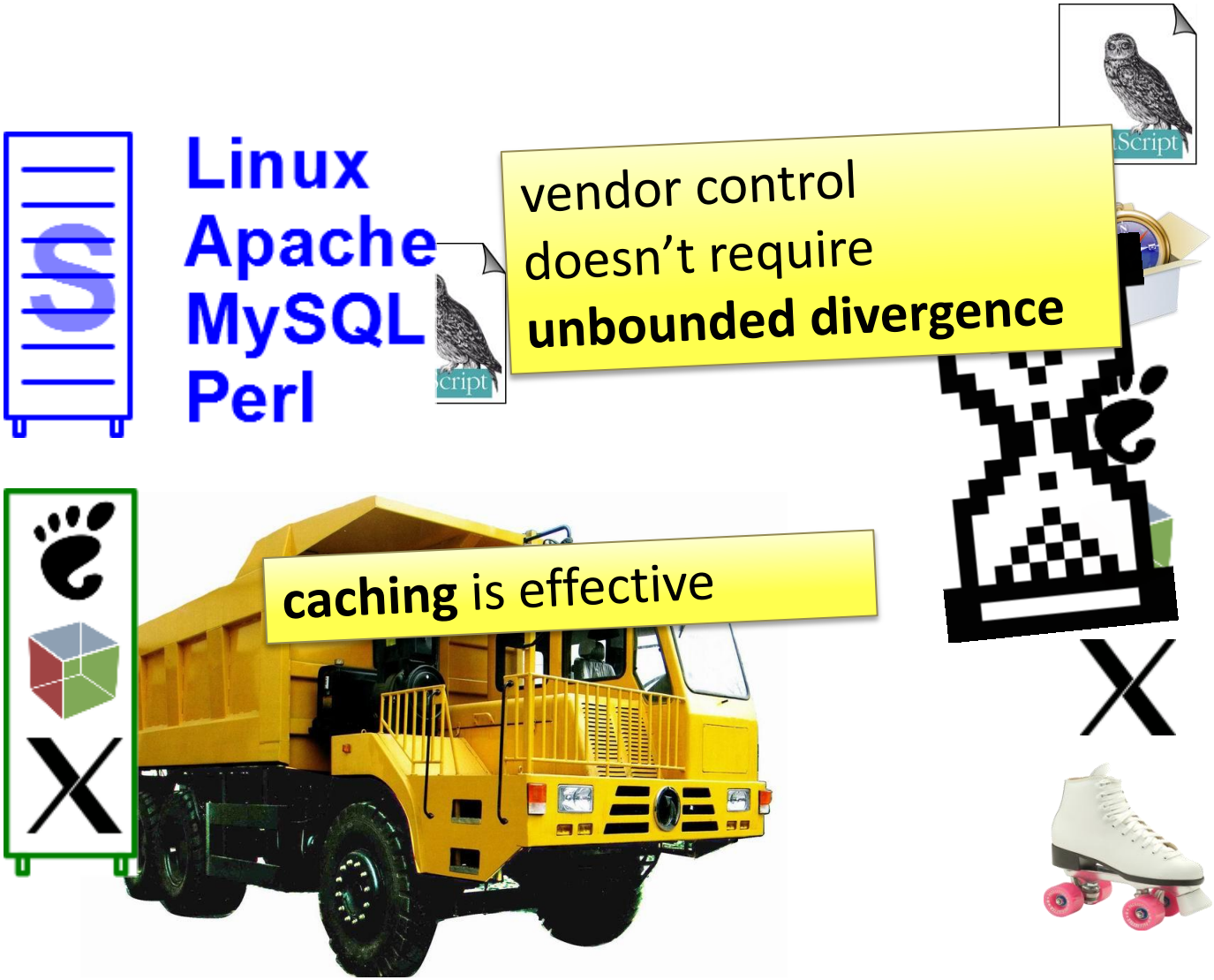
apps coordinate UI
using protocols

interaction: **Embassies** page navigation

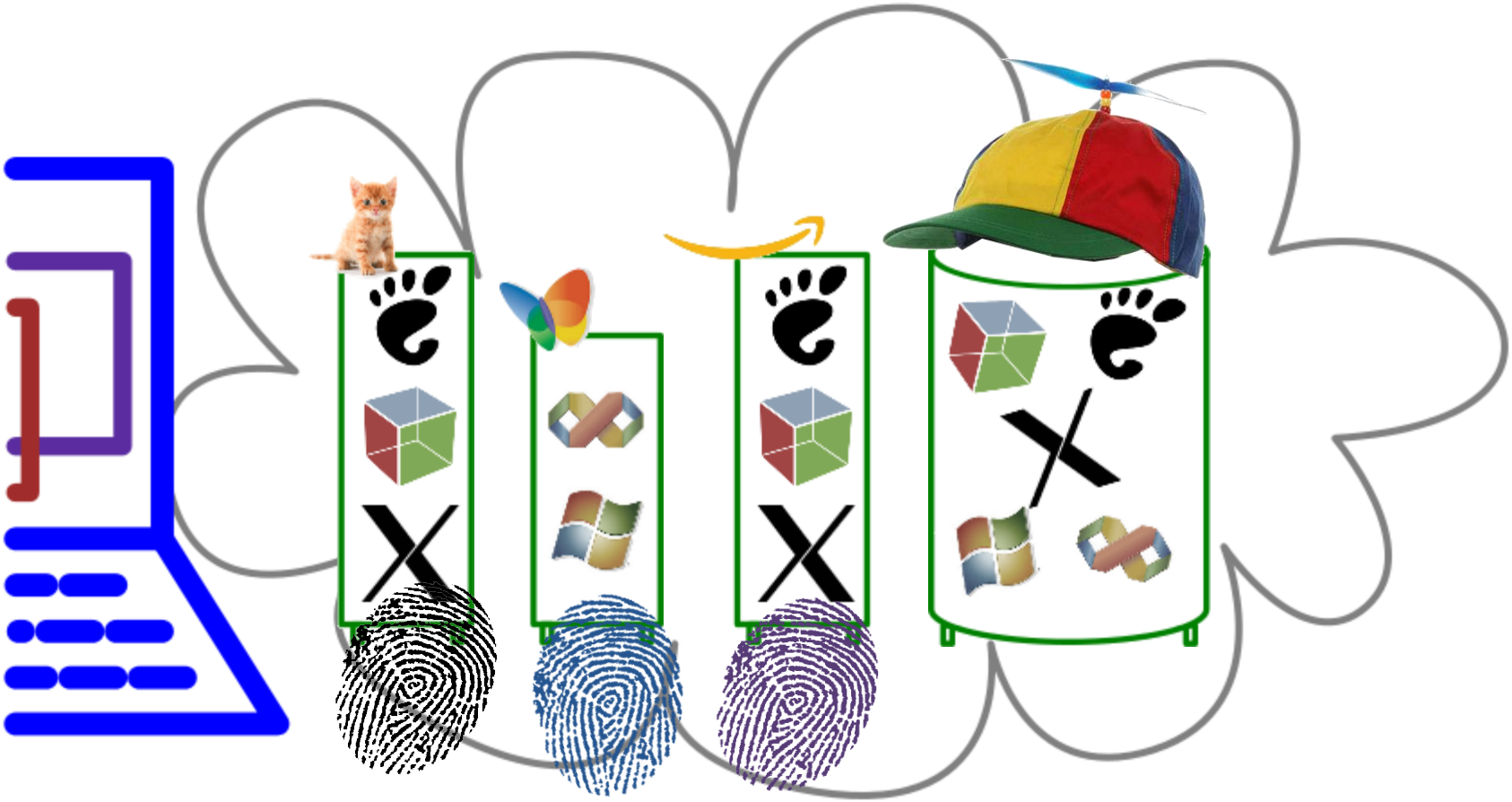


protocols make it clear how leaky an interaction is.

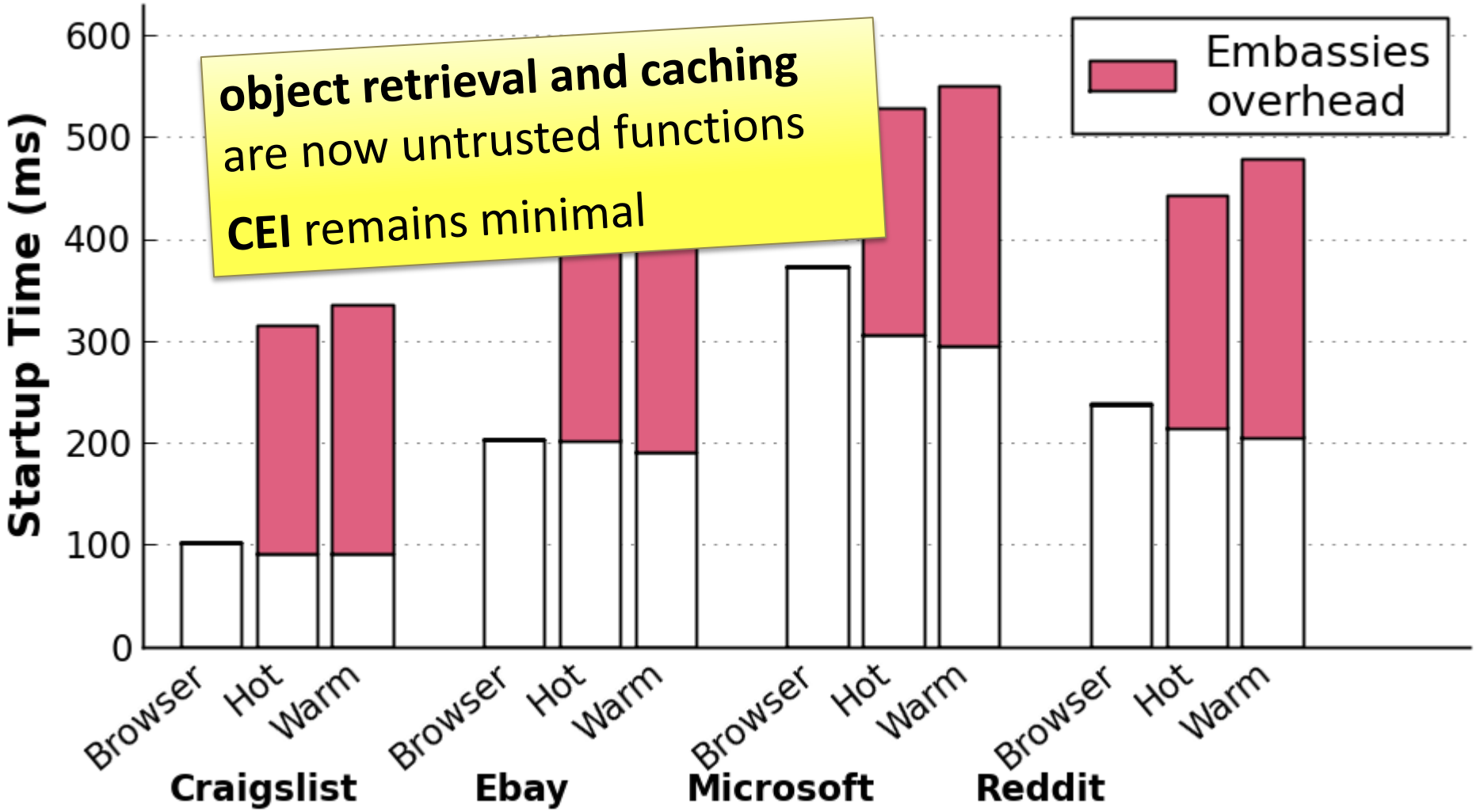
challenge: app launch performance



solution: untrusted cache



startup caching is effective



isn’t 200 ms a lot?

we’re only adding it when the user crosses over to a new site.

- within a site, vendors can go *faster*: SPDY++?

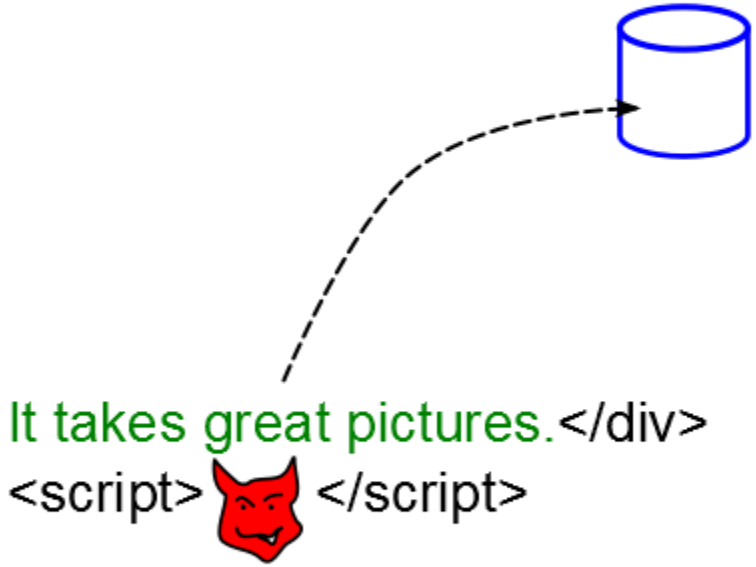
we’re loading unoptimized WebKit

this **modest performance problem**

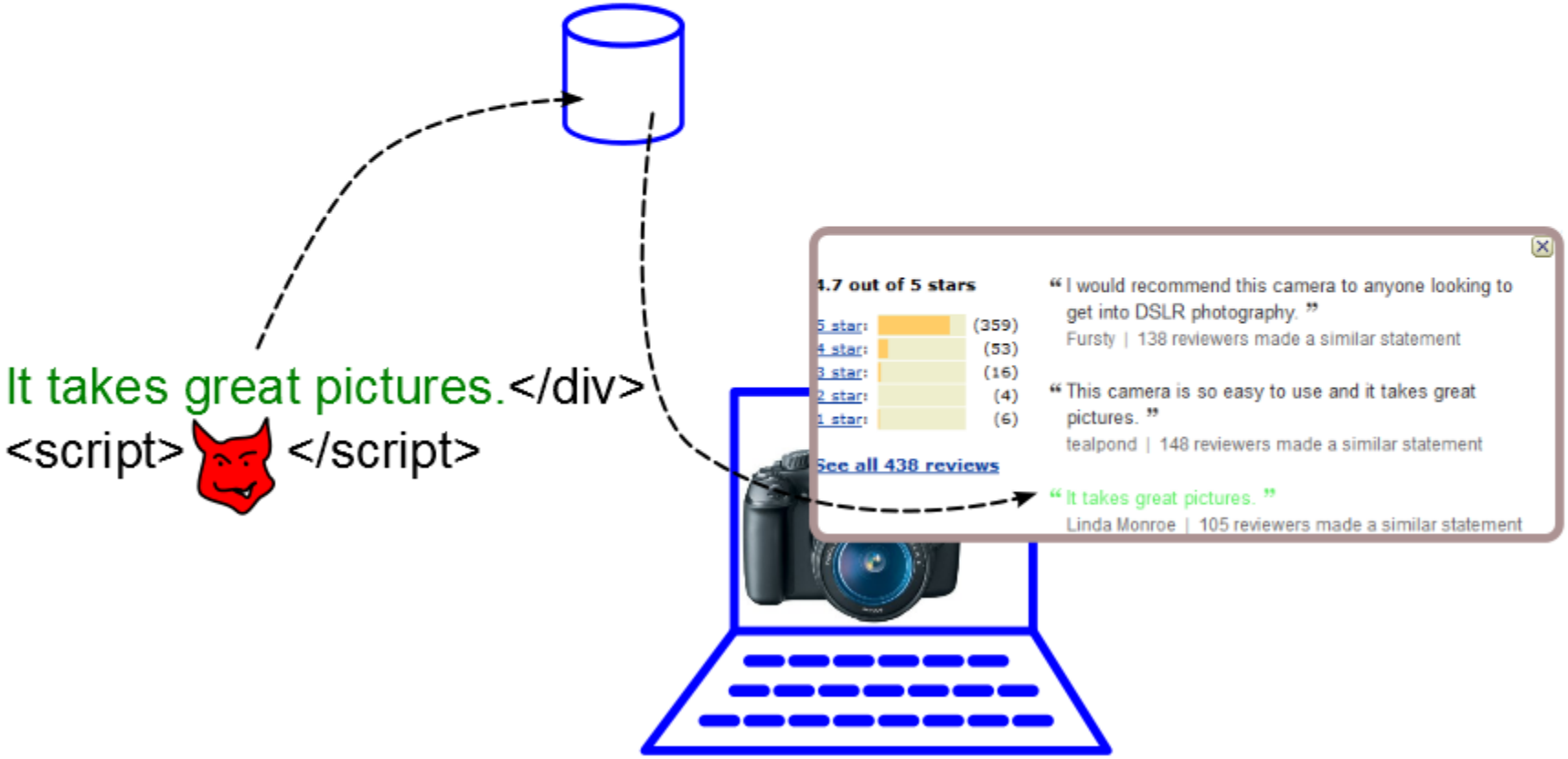
resolves a **bucket of security problems**

fixing flaws: history leaks

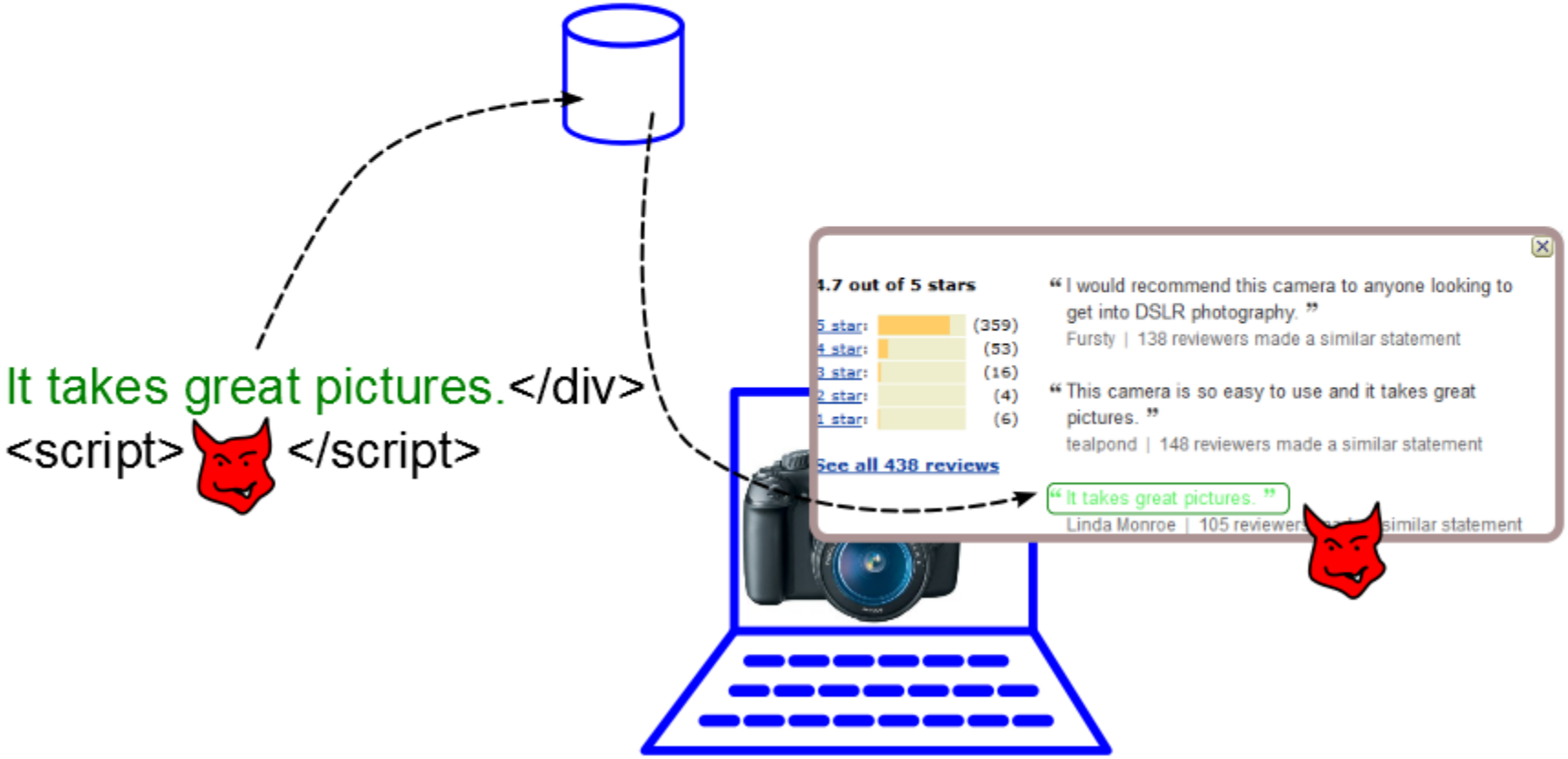
fixing flaws:
cross-site scripting (XSS)



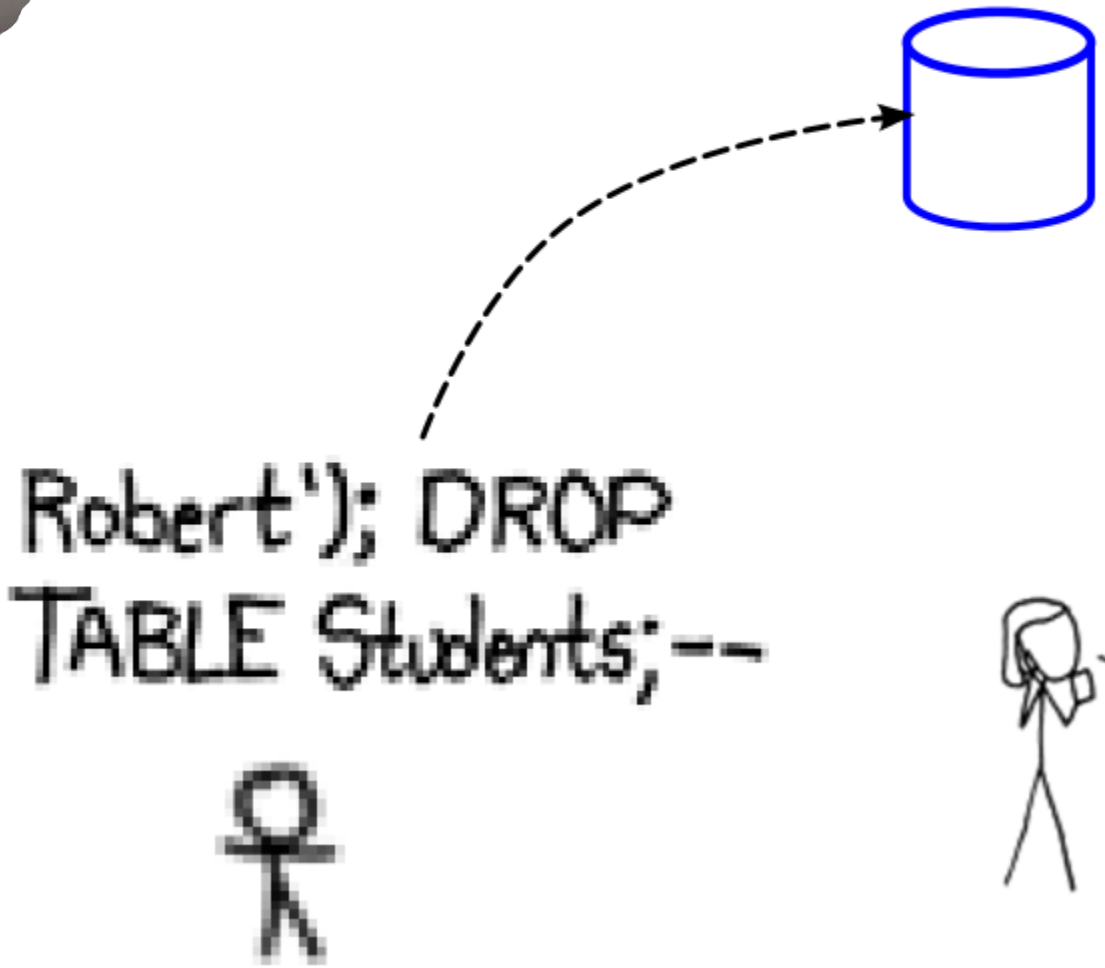
fixing flaws: cross-site scripting (XSS)



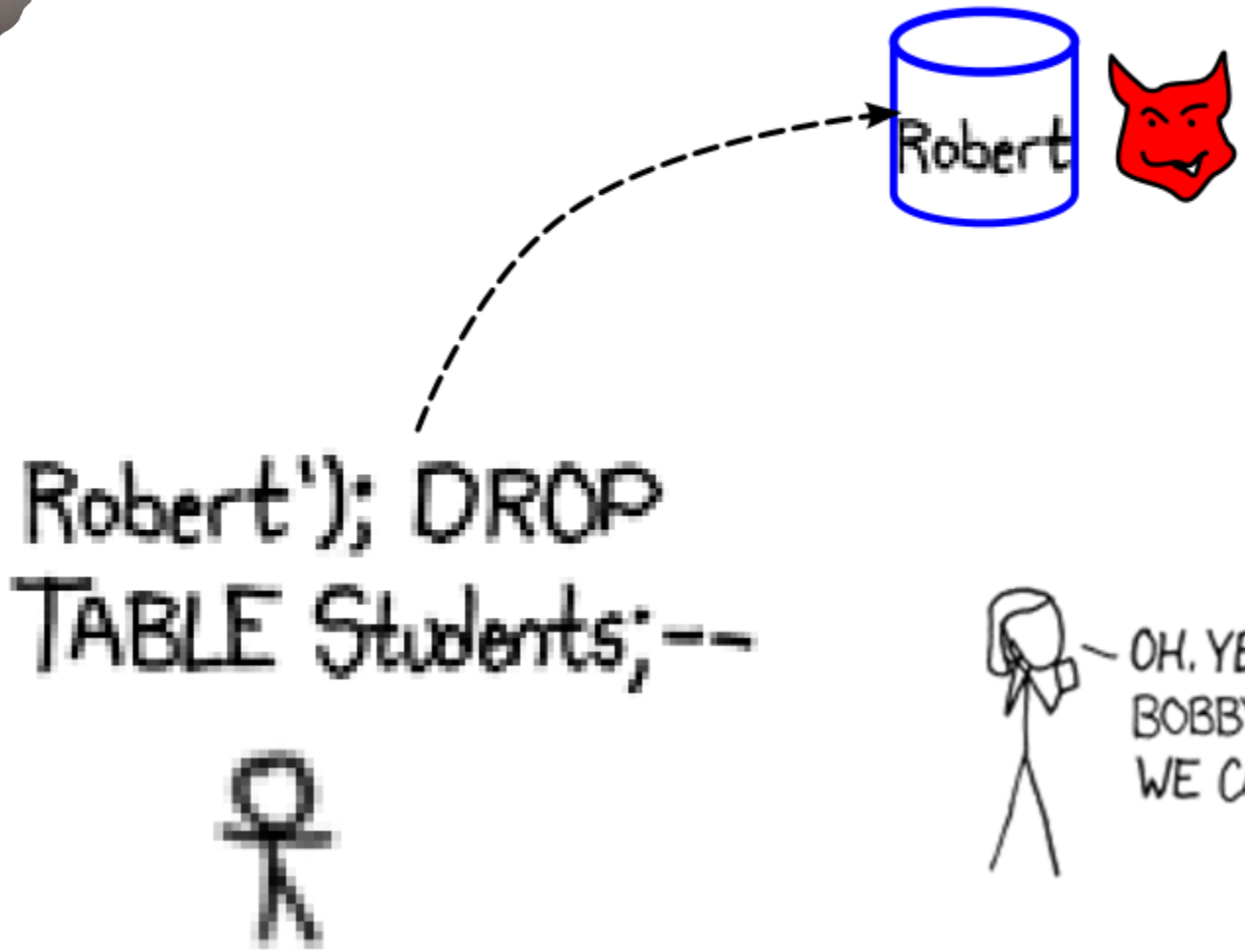
fixing flaws: cross-site scripting (XSS)



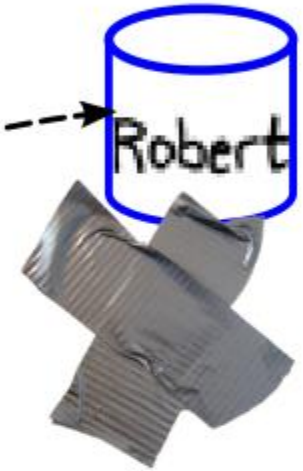
server analogue: SQL injection



server analogue: SQL injection

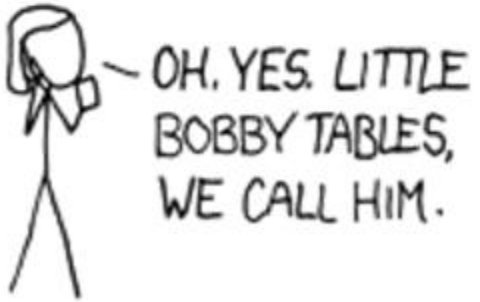


server analogue: SQL injection



vendors fix their own **servers**

Robert'); DROP TABLE Students;--



fixing flaws: cross-site scripting (XSS)

vendors fix their own **clients**

Summary

- The web API conflates CEI and DPI
- A **minimal** CEI can isolate correctly
- **native code** allows **rich** DPIs
- Launching big DPIs **isn't cost-prohibitive**
- The **pico-datacenter analogy** makes security tradeoffs obvious

No more dangerous links!

research.microsoft.com/embassies/

- linux & microkernel clients
- Webkit with protocol communication
- Gimp, Inkscape, spreadsheet, word processor
- untrusted app cache

what about mashups and serendipitous interoperability?

- Today, servers speak open protocols like XML and JSON; we can scrape HTML

- A few standard stacks will use a few standard wire protocols
- Sure, adversarial vendors can obfuscate, but they can do that in JavaScript, too.

shouldn't *I* control my browser?

- Shouldn't *I* get to control my browser?
  - ad blocker

- Letting a user give a third-party program (or plugin) full authority opposes vendor autonomy
  - Trojans / drive-bys
  - Autonomy means vendors can provide a predictable, safe experience

Accessibility

vendor control
doesn't require
unbounded divergence

Popular stacks (e.g. Windows, Gnome) include accessibility affordances.

Cross-architecture compatibility

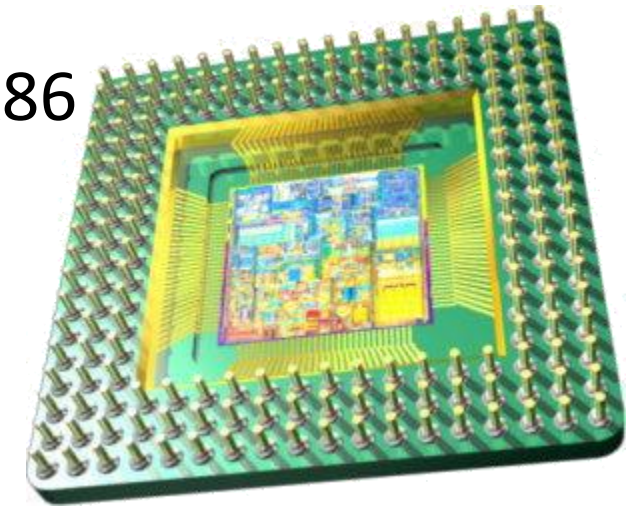
Three approaches:

- Managed code (JS, Java, C#) still a fine plan

  just deploy it from the vendor

- Cross-compile. Debian runs on a dozen archs.
- Binary rewriting

  got Apple from 68K to PowerPC to x86



Peripherals

- Printers *already* speak IP

  Google Cloud Print “IP-ifies” your legacy printer
- Same approach for GPS, cameras…
- Disks are easy

  untrusted “Seagate” app exposes storage

GPUs

- Long term:

  treat GPU like CPU

- Intermediate:

  exploit GPU segmentation as memory protection

- Near term:

  Even native CPU is pretty sweet

Deployment

- Start with a browser plug-in

  users enjoy rich apps, like NaCl

- Embassies client with compatibility mode

  supply a default DPI for “legacy” sites;

  Embassies-aware sites explicitly disable legacy mode