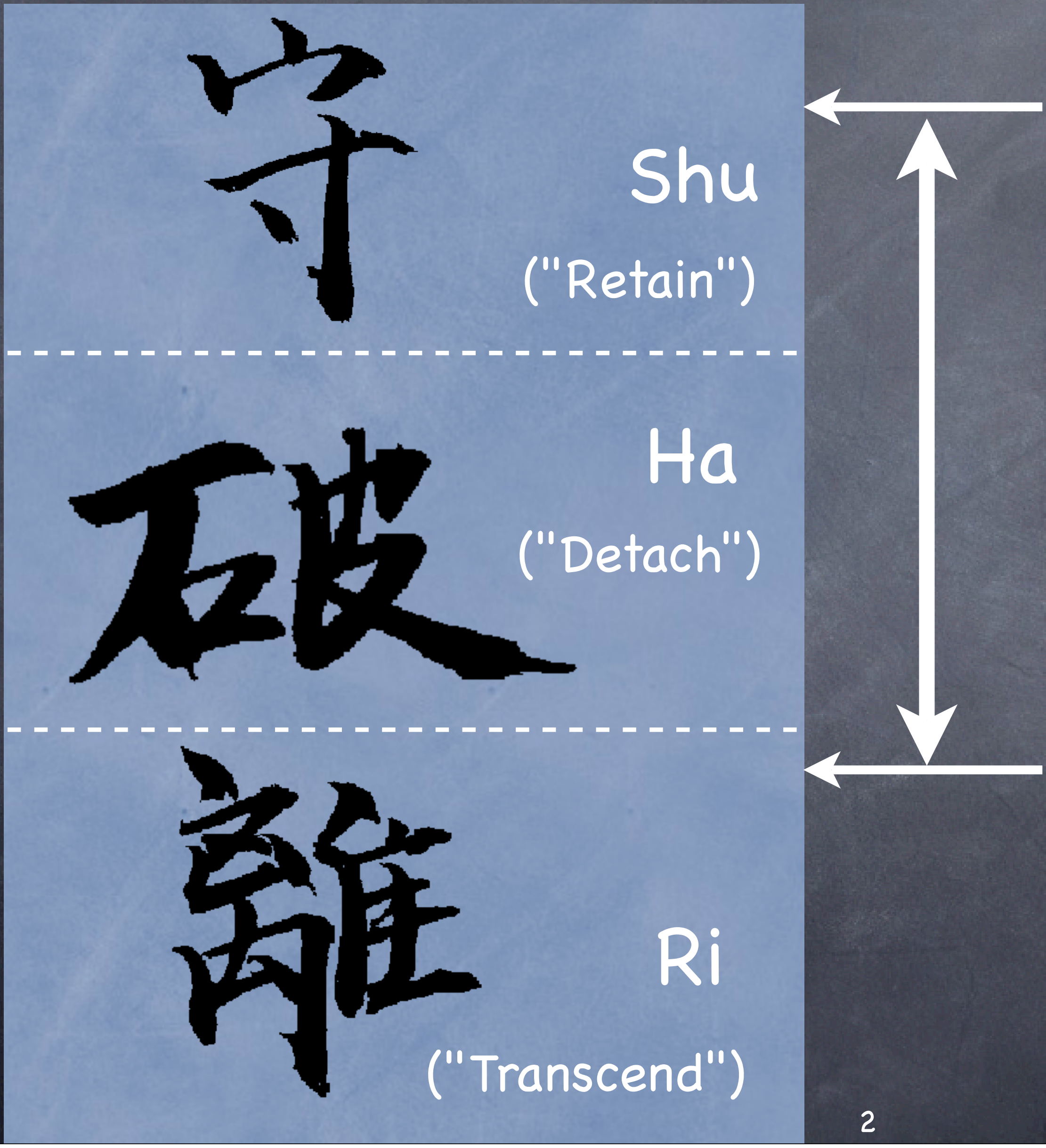
Abstraction as Leverage

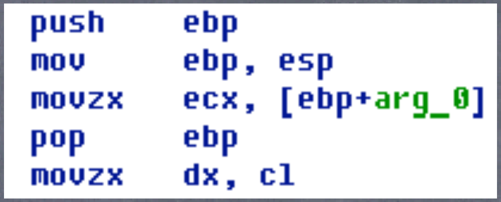
http://www.aleax.it/pycon_abst.pdf

Audience levels for this talk

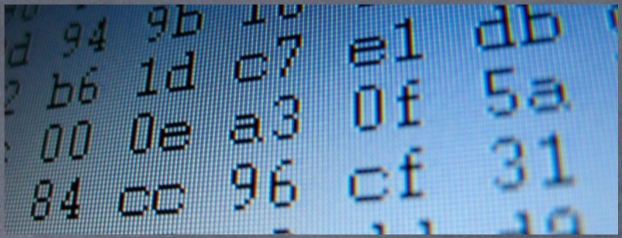


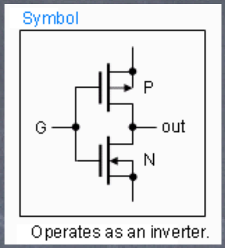
Q & A at the end please!

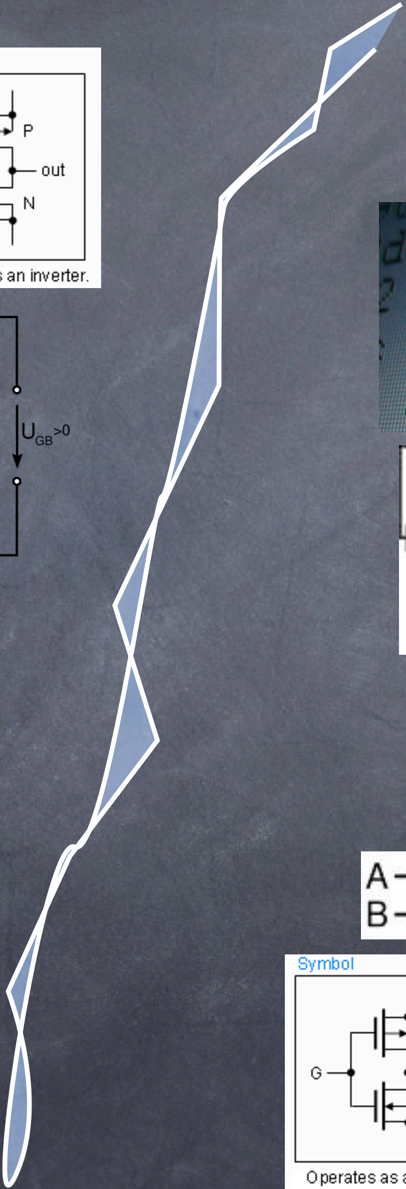
A Tower of Abstraction

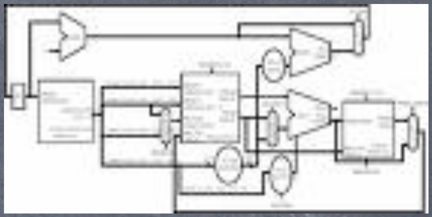


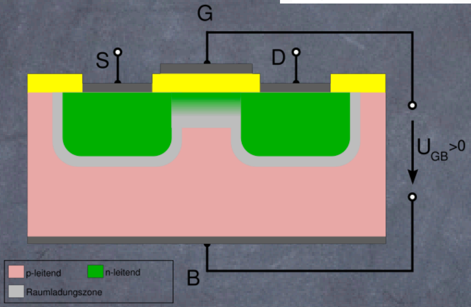
```
push    ebp
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
```

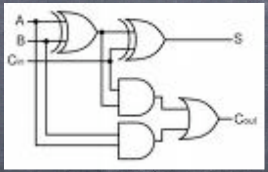


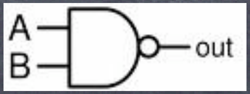


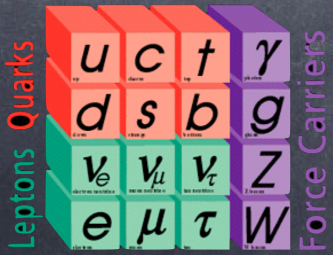


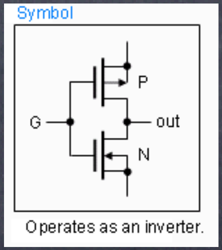






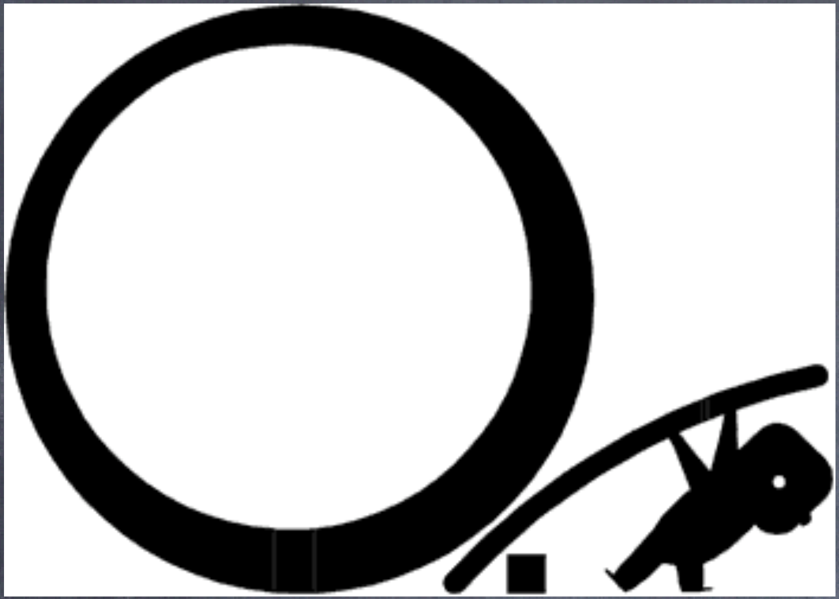






$$-\sum_i \frac{\hbar^2 \nabla_i^2 \Psi}{2m_i} + \sum_{i>j} V_{i,j}\Psi = E\Psi$$

Leverage…



…lets you do much more with so little…

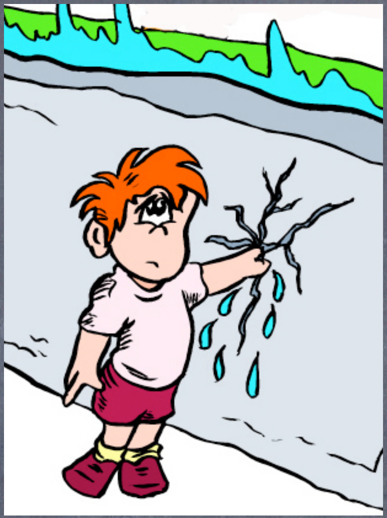
…but can crush you if things go wrong!

Can't live without it...

- programming (& most other "knowledge work")
  - always USES layers of abstraction,
  - often PRODUCES new layers on top

...can't live with it???

- all abstractions "LEAK" (Spolsky's Law)



  - bugs, overloads, security attacks, ...
  - ... so you MUST "get" some levels below!
- plus, abstractions SHOULD (s.t.;-) "leak"
  - in a designed, architected fashion

and: abstraction *can slow you down*!

Abstract -> Procrastinate!

- McCrea, S. M., Liberman, N., Trope, Y., & Sherman, S. J. -- Construal level and procrastination. Psychological Science, Volume 19, Number 12, December 2008, pp. 1308-1314(7)
- events remote in time are represented more abstractly than ones that are close in time
- McCrea et al. empirically prove the reverse also holds: more-abstract construal levels lead to higher likelihood of procrastination
- (at least for psych students - the only experimental subjects in ALL literature;-)
- Also: http://www.codinghorror.com/blog/archives/000922.html

Google

To achieve, think CONCRETE

- Allen's "Getting Things Done":
  - what's my SINGLE NEXT ACTION?
- *Personas* in interaction design (and user-centered design):
  - NOT "the user", BUT "Joe Blow, an inexperienced trader with lots of videogame experience, ..." or "Marc Smith, a seasoned trader who started back in the time of Hammurabi and is STILL most comfortable with cuneiform, ..."
- "prefer action over abstr-action" (J. Fried, "37 signals" founder)

The Abstraction Penalty

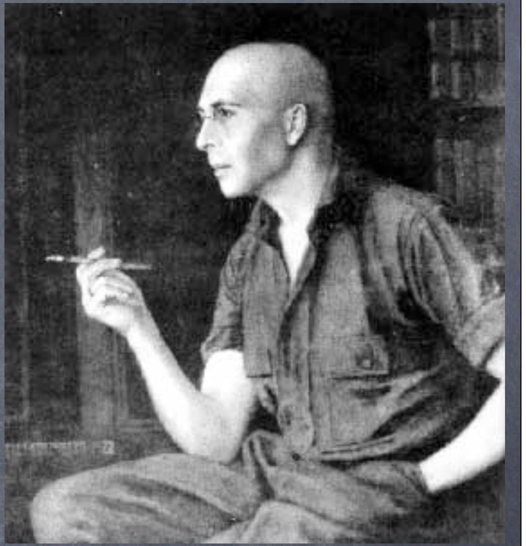
- when a language affords low-abstraction AND high-abstraction approaches, there may be a penalty for abstraction: cfr Stepanov's paper and benchmark at http://std.dkuug.dk/JTC1/SC22/WG21/docs/PDTR18015.pdf (& much further research)
- an issue of quality of implementation, and not always the case: in Python we're more used to get an *Abstraction Bonus* than any abstraction penalty;-)
- often via itertools, but not necessarily...

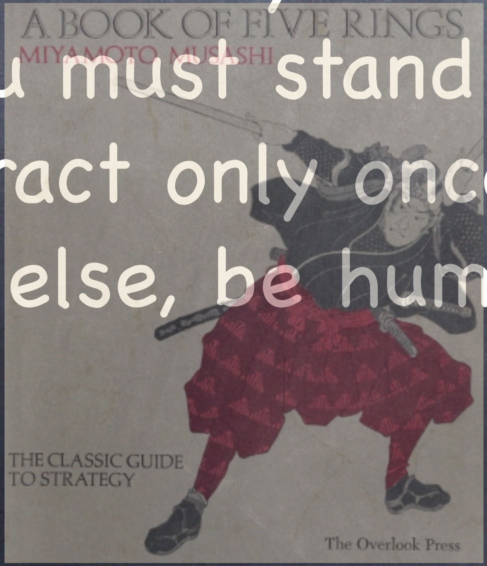
An Abstraction Bonus

```
$ python -mtimeit -s'x="abracadabra"' \
> 'y="".join(reversed(x))'
100000 loops, best of 3: 5.96 usec per loop
$ python -mtimeit -s'x="abracadabra"' \
> 'y=x[::-1]'
1000000 loops, best of 3: 0.597 usec per loop
```

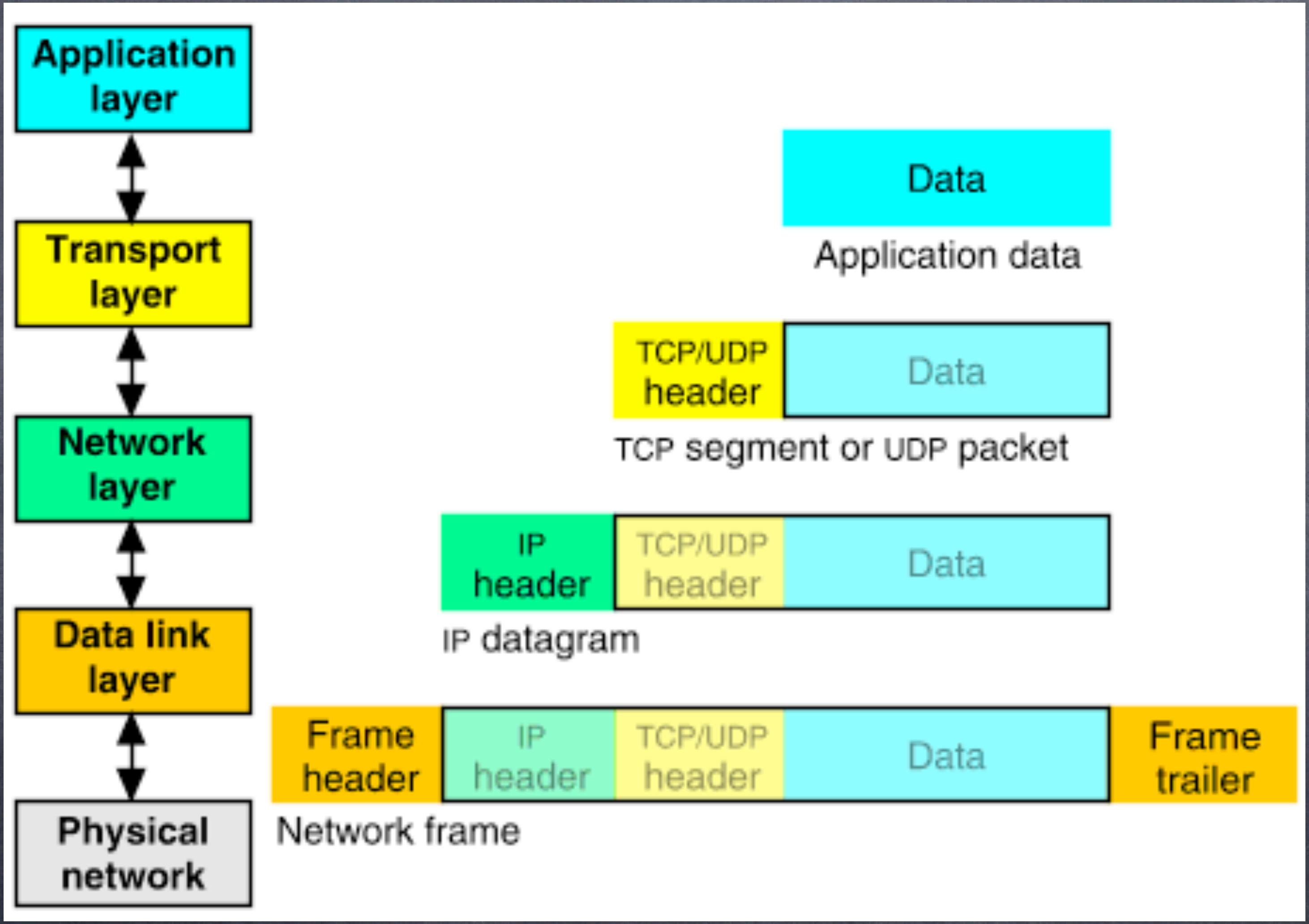
All Abstractions Leak

- all abstractions leak, because...:
  - ...*all abstractions LIE*!
  - the map is not the territory
- before you can abstract,
  - you must see the details
- i.e.: before you can withdraw,
  - you must stand close
- abstract only once you know all the details
  - or else, be humble & flexible about it!





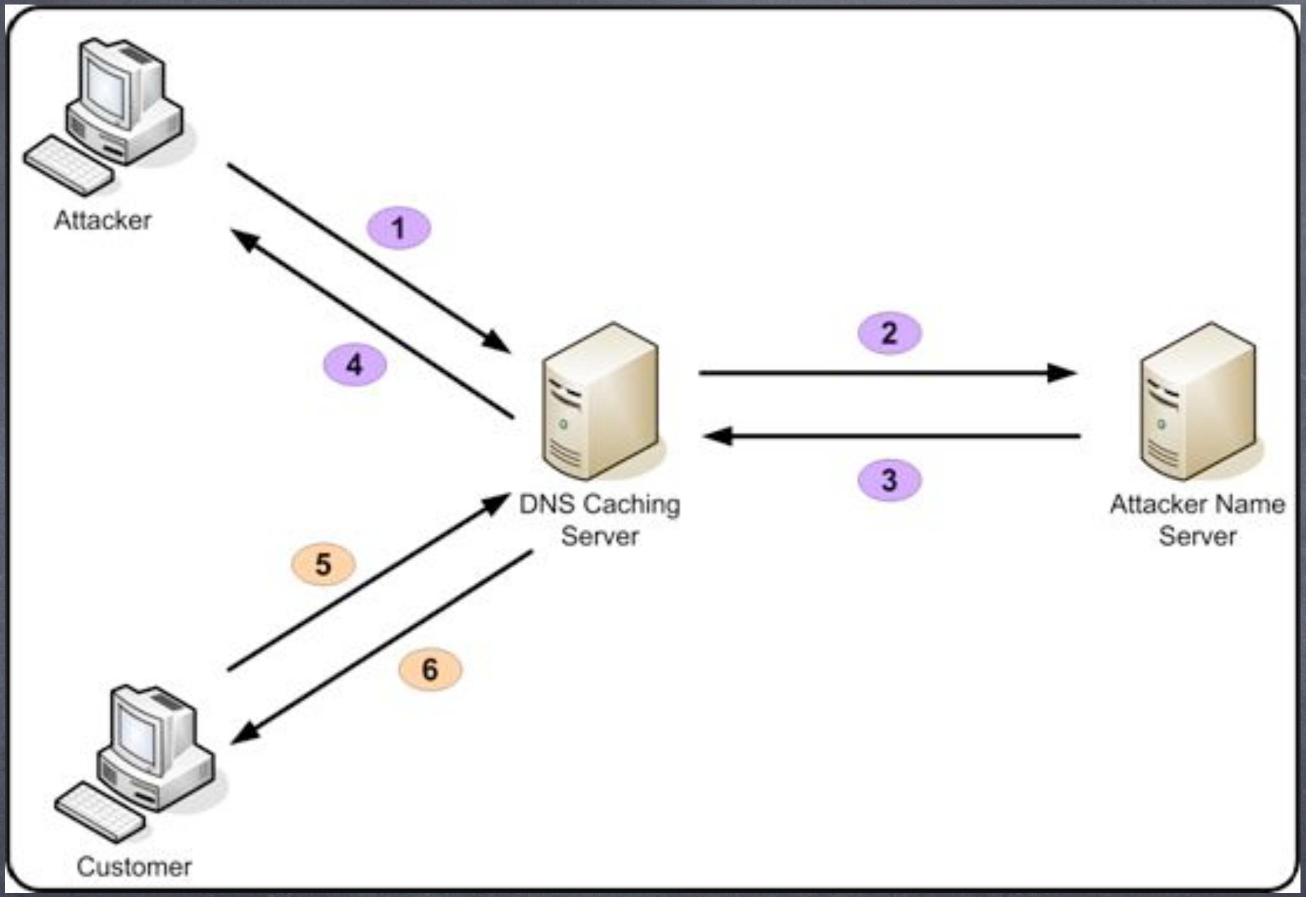
A great example: TCP/IP



TCP/IP's "leak": TRUST!

- TCP/IP's a great abstraction stack, BUT...
- ...it was designed in a long-ago era of trust!
- The whole stack "leaks" all over the place in terms of exposure to "sneaky" players
  - "below" (ARP cache poisoning),
  - "above" (DNS cache poisoning),
  - "to the sides" (BGP lies),
  - ...etc, etc...

One "leak": DNS Poisoning



&, some SHOULD leak!

- example: remote/distributed file systems
  - typically try to mimic "local" ones
  - the less local, the costlier the mimicry
    - local FS semantics, locking, reliability, ...
  - "filesystem" may be a superb abstraction
  - but "LOCAL filesystem" is definitely NOT!
  - ("never subclass concrete classes"...)
- doesn't mean the abstraction's BAD to have
  - but you can't have ONLY the abstraction!
  - need systematic ways to get "below" it

Good Abstraction Use

- you MUST be fully aware of at least a couple of layers “below”
- and to DESIGN an excellent abstraction:
  - be VERY familiar with SEVERAL expected implementations (“layers below”)
  - be VERY familiar with SEVERAL expected uses (“layers above”)
  - i.e.: no blinders, no shortcuts!
- YOU may be the next implementer OR user!
  - the Golden Rule makes EXTRA sense;-)
- http://c2.com/cgi/wiki?TooMuchAbstraction

A Donald Knuth quote

- the psychological profiling [[of the programmer]] is mostly the ability to shift levels of abstraction, from low level to high level. To see something in the small and to see something in the large. [[...]]
- Computer scientists see things simultaneously at the low level and the high level [[of abstraction]]

http://www.ddj.com/184409858

A Jason Fried quote

- "Here's the problem with copying:
  - Copying skips understanding.
  - Understanding is how you grow.
  - You have to understand why something works or why something is how it is.
  - When you copy it, you miss that.
  - You just repurpose the last layer instead of understanding the layers underneath."
- Just '%s/copy/use existing high-level abstractions blindly/g' ...;-)

http://www.37signals.com/svn/posts/
1561-why-you-shouldnt-copy-us-or-anyone-else

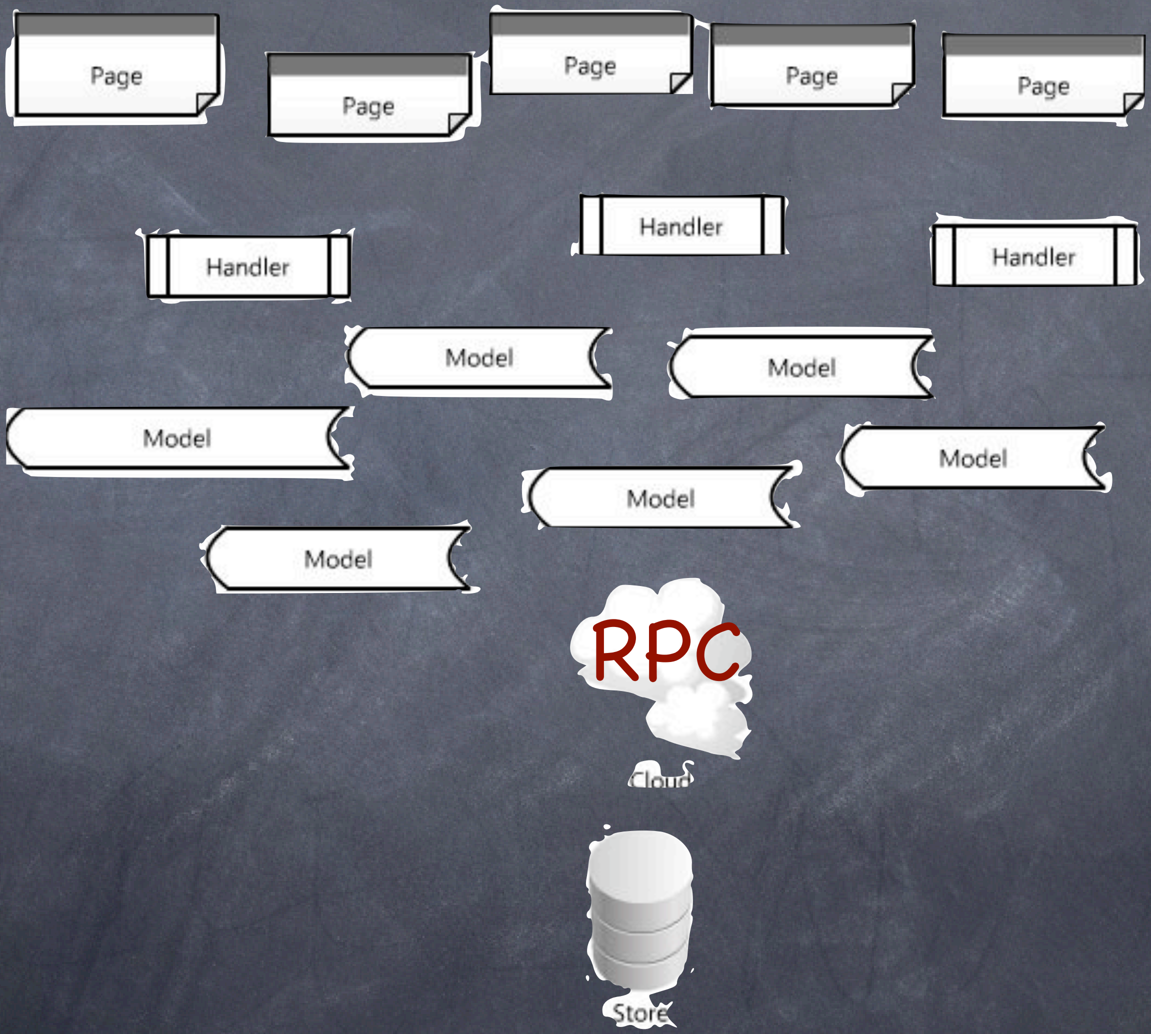
A Jeff Atwood quote

- "don't reinvent the wheel,
  - unless you plan on learning more about wheels!"

http://www.codinghorror.com/blog/archives/001145.html

Google

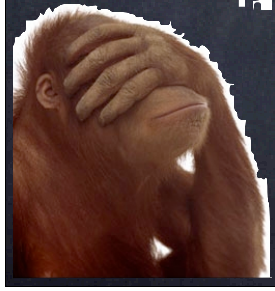
App Engine "hacks"



The monkeypatching way

- all operations go through an RPC layer, via `apiproxy_stub_map.MakeSyncCall`
- the wrong answer: *monkey-patch* it...:

```
from google.appengine.api import \
    apiproxy_stub_map
_org = apiproxy_stub_map.MakeSyncCall
def fake(svc, cal, req, rsp):
  ...
  x = _org(svc, cal, req, rsp)
  ...
apiproxy_stub_map.MakeSyncCall = fake
```



Better answer: HOOKS

see: http://blog.appenginefan.com/2009/01/hacking-google-app-engine-part-1.html

```
from google.appengine.api import apiproxy_stub_map
def prehook(svc, cal, req, rsp):
  ...
apiproxy_stub_map.apiproxy.GetPreCallHooks(
  ).Append('unique_name', prehook, 'opt_api_id')
```

Q & A

http://www.aleax.it/pycon_abst.pdf

?

!